robotron

Programmtechnische Beschreibung

# REDABAS-4

## Relationales Datenbankbetriebssystem

Teil 1

VEB ROBOTRON · PROJEKT DRESDEN

H1414-2051-1 M3030

DCP

Programmtechnische Beschreibung

# REDABAS-4

Relationales Datenbankbetriebssystem

Teil 1


**VEB Robotron-Projekt Dresden**

H 1414-2051-1 M3030

Die vorliegende Dokumentation Programmtechnische Beschreibung entspricht dem Stand von 07/89.

Nachdruck, jegliche Vervielfaeltigung oder Auszuege daraus sind unzulaessig.

Die Ausarbeitung erfolgte durch ein Kollektiv des VEB Robotron-Projekt Dresden.

Herausgeber: VEB Robotron-Projekt Dresden
Leningrader Str. 9
Dresden
8010

Kurzreferat
---

REDABAS-4 DCP ist fuer den Einsatz auf dem Arbeitsplatzcomputer A 7150 mit dem Betriebssystem DCP 1700 und dem Personalcomputer EC 1834 mit dem Betriebssystem DCP ab Version 3.20 im Einzelnutzerbetrieb vorgesehen.

REDABAS-4 DCP kann ausserdem auf kompatiblen Anlagen mit entsprechenden Betriebssystemen eingesetzt werden.

Die vorliegende Programmtechnische Beschreibung enthaelt detaillierte Angaben ueber die Arbeitsweise, die Funktionen der Datenbankverwaltung, das Programmieren und die Datenorganisation von REDABAS-4 DCP.

Eine ausfuehrliche Beschreibung der REDABAS-4-Funktionen und -Befehle bietet dem Anwender in Verbindung mit praxisnahen Demonstrationsbeispielen exakte Mittel fuer die Handhabung und Nutzung von REDABAS-4 DCP.

Die vorliegende Dokumentation ist fuer REDABAS-4 1.0 DCP und folgende Ausgaben gueltig, sofern nicht durch Aenderungsdienst und nachfolgende Schriften anderes angezeigt wird.

Ag706/H/664/89 III/9/319

Inhaltsverzeichnis Seite

Teil 1


Einfuehrung ..... 9

1. Vergleich der Leistungsparameter von REDABAS-4 mit Vorgaengersystemen ..... 9
1.1. Vergleich quantitativer Leistungsparameter ..... 10
1.2. Vergleich qualitativer Leistungsparameter ..... 11
1.2.1. Vergleich REDABAS - REDABAS-4 ..... 11
1.2.2. Vergleich REDABAS-3 - REDABAS-4 ..... 12

2. Arbeitsweise ..... 12
2.1. Voraussetzungen fuer die Anwendung ..... 12
2.2. Allgemeine Hinweise ..... 13
2.3. Inbetriebnahme von REDABAS-4 ..... 14
2.3.1. Was Sie vor dem ersten Start von REDABAS-4 tun muessen ..... 14
2.3.2. Der REDABAS-4-Start ..... 16
2.3.3. Das Beenden ..... 18
2.4. Die Hilfsmoeglichkeiten ASSIST und HELP ..... 19
2.5. Betriebsarten und Aufteilung des Bildschirms ..... 21

3. Funktionen der Datenbankverwaltung ..... 23
3.1. Einrichten und Aendern von Datenbankdateien ..... 23
3.1.1. Erstellen einer Datenbankdatei ..... 23
3.1.2. Eingeben weiterer Datensaetze ..... 31
3.1.3. Aendern von Datensaetzen ..... 34
3.1.4. Loeschen von Datensaetzen ..... 37
3.2. Wiederauffinden aus der Datenbank ..... 39
3.2.1. Selektion bestimmter Datenfelder ..... 39
3.2.2. Bedingte Auflistung von Datensaetzen ..... 40
3.2.3. Das Ausblenden von Datensaetzen ..... 45
3.2.4. Bedingte Suche einzelner Datensaetze ..... 47
3.2.5. Bedingte Anzeige von Datensaetzen ..... 48
3.2.6. Positionieren auf Datensaetze ..... 48
3.2.7. Abfragen einzelner Datenfelder ..... 49
3.3. Aendern der Dateistruktur ..... 51
3.4. Sortieren und Indizieren ..... 54
3.4.1. Sortieren ..... 54
3.4.2. Indizieren ..... 57
3.4.3. Wiederauffinden mittels Indexdateien ..... 58
3.5. Auswerten und Verdichten ..... 60
3.5.1. Summenbildung ..... 61
3.5.2. Zaehlen von Datensaetzen ..... 62
3.5.3. Mittelwertbildung ..... 63
3.5.4. Verwendung von Speichervariablen ..... 63
3.5.5. Verdichten von Dateien ..... 65
3.6. Die Arbeit mit Merkfeldern ..... 66
3.7. Reportgenerator ..... 72
3.8. Etikettengenerator ..... 87
3.9. Querygenertor ..... 91
3.10. Maskengenerator ..... 95
3.11. Gleichzeitiges Arbeiten mit mehreren Datenbankdateien ..... 103
3.11.1. Arbeitsbereiche und Aliasnamen ..... 103
3.11.2. Gleichzeitiges Bearbeiten von Datenbankdateien mit unterschiedlicher Dateistruktur ..... 104
3.11.3. Verknuepfen von zwei Datenbankdateien ..... 110

3.11.4. Viewgenerator ..... 111
3.12. Zusammenfassung ..... 118

4. Programmieren ..... 119
4.1. Befehlsfolgen ..... 119
4.2. Programmschleifen ..... 128
4.3. Programmverzweigungen ..... 132
4.4. Prozeduren ..... 137
4.5. Programmierte Dateneingabe ..... 139
4.6. Zusammenfassung ..... 141

5. Das Aufrufen externer Programme ..... 143
5.1. Moeglichkeiten des RUN-Befehls ..... 143
5.2. Ausfuehren externer Programme mit CALL ..... 144
5.3. Parameteruebergabe ..... 145
5.4. Datenaustausch mit externen Programmen ..... 146
5.5. Ergaenzende Hinweise ..... 149

6. Die Benutzung von Katalogdateien ..... 150
6.1. Erstellen, Oeffnen und Schliessen einer Katalogdatei ..... 150
6.2. Automatische Aktualisierung von Eintragungen in einer Katalogdatei ..... 153
6.3. Die Struktur von Katalogdateien ..... 154
6.4. Manipulation in Katalogdateien ..... 155
6.5. Die Katalog-Frage-Klausel ..... 156

7. Umsteigen von REDABAS auf REDABAS-4 ..... 159
7.1. Transformation von Datenbankdateien ( .DBD) ..... 160
7.2. Transformation von Indexdateien ( .IDX) ..... 160
7.3. Transformation von Programm- und Maskendateien ( .PRG, .MSK) ..... 160
7.4. Transformation von Speichervariablen- und Reportdateien ( .VAR, .DEF) ..... 166

8. Allgemeine Informationen ..... 167
8.1. Datenorganisation ..... 167
8.1.1. Aufbau der Datenbank ..... 167
8.1.2. Datentypen und Feldformate ..... 168
8.1.2.1. Datentypen ..... 168
8.1.2.2. Feldformate ..... 171
8.1.3. REDABAS-4-Dateien ..... 174
8.1.3.1. Datenbankdateien ( .DBD) ..... 176
8.1.3.2. Katalogdateien ( .CAT) ..... 176
8.1.3.3. Merkdateien ( .DBM) ..... 177
8.1.3.4. Sicherungsmerkdateien ( .TBK) ..... 177
8.1.3.5. Indexdateien ( .IDX) ..... 177
8.1.3.6. Speichervariablendateien ( .VAR) ..... 178
8.1.3.7. Programmdateien ( .PRG) ..... 178
8.1.3.8. Reportdateien ( .DEF) ..... 178
8.1.3.9. Etikettendateien ( .ETI) ..... 179
8.1.3.10. Querydateien ( .QRY) ..... 179
8.1.3.11. Screndateien ( .SCR) ..... 179
8.1.3.12. Maskendateien ( .MSK) ..... 179
8.1.3.13. Viewdateien ( .VUE) ..... 180
8.1.3.14. Textdateien ( .TXT) ..... 181
8.1.3.15. Sicherungsdateien ( .BAK) ..... 181
8.2. Speichervariablen ..... 182
8.2.1. Ueberblick ..... 182
8.2.2. Datentypen von Speichervariablen ..... 183
8.2.3. Speichervariablen-Verwaltung ..... 184

8.2.4. Zuweisungen ... 185
8.2.5. Existenzbereich und Sichtbarkeit ... 187
8.2.6. Namenskonflikte zwischen Speichervariablen und Feldern ... 190
8.2.7. Parameteruebergabe und Ergebnisrueckmeldung ... 191
8.2.8. Allgemein verwendbare Programme ... 193
8.3. Befehlssyntax und Ausdruecke ... 196
8.3.1. Syntax ... 196
8.3.1.1. Syntax der Befehle ... 196
8.3.1.2. Syntax der Ausdruecke ... 197
8.3.2. Variablen und Konstanten ... 199
8.3.2.1. Variablen ... 199
8.3.2.2. Konstanten ... 199
8.3.3. Operationen ... 200
8.3.3.1. Arithmetische Operationen (Berechnungen) ... 202
8.3.3.2. Vergleichsoperationen ... 203
8.3.3.3. Logische Operationen ... 205
8.3.3.4. Zeichenreihen-Operationen ... 206
8.3.3.5. Datumsoperationen ... 207
8.3.3.6. Rangfolge von Operationen ... 207
8.4. Hinweise zur Dialogfuehrung ... 209
8.4.1. Fehlerbehandlung ... 209
8.4.1.1. Interaktiver Modus ... 209
8.4.1.2. Programmodus ... 210
8.4.2. Seitenmodus und Cursorsteuerung ... 212
8.5. Zugriffspfade ... 216
8.5.1. Aufruf der REDABAS-4-Systemkomponenten ... 217
8.5.2. Dateizugriff in REDABAS-4 ... 218
8.5.3. Beispiele zu den Arbeitsmoeglichkeiten ... 220
8.6. Uebersichten zur Datensatzzeigerverwaltung ... 223
8.6.1. Bedingungen bei nicht-indizierten Datenbankdateien ... 223
8.6.2. Bedingungen bei indizierten Datenbankdateien ... 224
8.6.3. Bedingungen bei SET DELETED OFF/ON ... 225

Teil 2
------

9. Funktionen und Befehle ... 226
9.1. Symboldefinitionen ... 226
9.2. Funktionen ... 231
9.3. Befehle ... 282
9.3.1. Uebersicht ... 282
9.3.2. Datensatzauswahl ... 291
9.3.3. Beschreibung der Befehle ... 293
9.4. Makros ... 446

10. REDABAS-4-Umgebung ... 449
10.1. Das Konfigurieren von REDABAS-4 ... 449
10.1.1. Ueberblick ... 449
10.1.2. Parameter, die mit SET-Befehlen aenderbar sind ... 450
10.1.3. Parameter, die nicht durch Befehle modifizierbar sind ... 453
10.2. Umkodieren der Druckausgabe ... 455
10.3. Schnittstellen ... 456

Anhang A: ... 460

Abkuerzungsverzeichnis ... 471

Sachwortverzeichnis ... 472

Bildverzeichnis


| | | | Seite |
|---|---|---|---|
| Bild | 1 | Die Arbeit mit dem ASSIST-Befehl | 20 |
| | 2 | HELP-Informationen im ASSIST-Modus | 21 |
| | 3 | Aufteilung des Bildschirms | 22 |
| | 4 | Eingabemaske fuer die Definition der Struktur einer Datenbankdatei | 25 |
| | 5 | Struktur der Datenbankdatei "kunden" | 28 |
| | 6 | Maske fuer die Datenerfassung | 29 |
| | 7 | Liste der Datenbankdatei "kunden" | 30 |
| | 8 | Struktur der Datenbankdatei "kunden" | 31 |
| | 9 | Maske fuer das Anfuegen eines neuen Datensatzes | 32 |
| | 10 | Erweiterte Datenbankdatei "kunden" | 32 |
| | 11 | (Unveraenderter) Datensatz Nr. 4 | 34 |
| | 12 | Veraenderter Datensatz Nr. 4 | 34 |
| | 13 | Datenbankdatei "kunden" nach der Aenderung | 35 |
| | 14 | Aendern eines Datensatzes mit CHANGE | 37 |
| | 15 | Datenbankdatei "kunden" mit den zur Loeschung markierten Datensaetzen | 38 |
| | 16 | Drei markierte Saetze der Datenbankdatei "kunden" sind physisch geloescht | 39 |
| | 17 | Datenbankdatei "kunden", spezifiziert nach Vorname, Name, Ort | 40 |
| | 18 | Datenbankdatei "kunden", in zwei Varianten durchsucht | 41 |
| | 19 | Datenbankdatei "kunden", mit Vergleichsoperator "=" durchsucht | 42 |
| | 20 | Datenbankdatei "kunden", mit Vergleichsoperator "<" durchsucht | 42 |
| | 21 | Demonstration des Teilzeichenreihen-Operators | 43 |
| | 22 | Anwenden des Teilzeichenreihen-Operators und der LOWER-Funktion | 44 |
| | 23 | Zwei mit "OR" verknuepfte Bedingungen | 44 |
| | 24 | Zwei mit "AND" verknuepfte Bedingungen | 44 |
| | 25 | Datenbankdatei "kunden", mit positioniertem Suchbegriff durchsucht | 45 |
| | 26 | Das Ausblenden von Datensaetzen | 46 |
| | 27 | Beispiel fuer die Anwendung der Befehle LOCATE, DISPLAY, CONTINUE | 47 |
| | 28 | Varianten des Befehls DISPLAY | 48 |
| | 29 | Varianten des Befehls GO | 49 |
| | 30 | Abfragen einzelner Felder mit dem Fragezeichen | 50 |
| | 31 | Neue Struktur der Datenbankdatei "kunden" | 52 |
| | 32 | Geaenderte Datenbankdatei "kunden" | 52 |
| | 33 | Durch den Befehl BROWSE mit der Kundennummer komplettierte Datenbankdatei "kunden" | 53 |
| | 34 | Datenbankdatei "kunden", sortiert nach der Kundennummer | 55 |
| | 35 | Datenbankdatei "kunden", sortiert nach Ort und Name | 56 |
| | 36 | Datenbankdatei "kunden", indiziert nach der Postleitzahl | 58 |
| | 37 | Struktur der Datenbankdatei "auftrag" | 60 |
| | 38 | Datenbankdatei "auftrag" | 60 |
| | 39 | Uebersicht der Speichervariablen | 64 |
| | 40 | Zusammenfassung der Auftragsdatei nach Kunden | 65 |
| | 41 | Anzeige der relevanten Felder der Datei "gesamt" | 66 |

| | | Seite |
|---|---|---|
| Bild 42 | Struktur der Datenbankdatei "kunden1" (mit Merkfeld) | 69 |
| 43 | Anzeige von Merkfeldern | 70 |
| 44 | Datenbankdatei "kunden1" | 71 |
| 45 | Struktur der Datenbankdatei "bestand" | 72 |
| 46 | Datenbankdatei "bestand" | 73 |
| 47 | Format-Menue des Reportgenerators | 74 |
| 48 | Format-Menue des Reportgenerators fuer Liste "Lagerbestand" | 75 |
| 49 | Gruppe-Menue des Reportgenerators | 75 |
| 50 | Spalte-Menue des Reportgenerators | 76 |
| 51 | Definition der ersten Spalte der Liste "Lagerbestand" | 77 |
| 52 | Eintragung der Spalte "Preis" | 78 |
| 53 | Mit dem Reportgenerator erstellter Lagerbestand (Reportdatei "best1") | 79 |
| 54 | Liste "Wertmaessiger Lagerbestand", nach Lagerorten aufgeschluesselt, mit Einzelpositionen (Reportdatei "best21") | 82 |
| 55 | Liste "Wertmaessiger Lagerbestand", nach Lagerorten aufgeschluesselt, nur Gruppensummen und Summe (Reportdatei "best22") | 84 |
| 56 | Liste "Ersatzteiluebersicht" (Reportdatei "best31") | 85 |
| 57 | Bedarfsliste (Reportdatei "best31") | 87 |
| 58 | Auswahl-Menue des Etikettengenerators (Auessere Gestaltung) | 88 |
| 59 | Inhalt-Menue des Etikettengenerators (Definition des Inhalts) | 89 |
| 60 | Ausdruck mit dem Etikettengenerator | 90 |
| 61 | Setze-Filter-Menue des Querygenerators | 92 |
| 62 | Anzeige-Menue des Querygenerators | 93 |
| 63 | Verknuepfung-Menue des Querygenerators | 94 |
| 64 | Individuelle Ein-/ Ausgabemaske fuer Datenbankdatei "kunden" | 96 |
| 65 | Aktionen zum Aufbau der Maskendatei "kundens1.msk" | 98 |
| 66 | Aufbau-Menue des Maskengenerators | 99 |
| 67 | Zeichenbrett des Maskengenerators | 99 |
| 68 | Aendern-Menue des Maskengenerators | 100 |
| 69 | Optionen-Menue des Maskengenerators | 101 |
| 70 | Ausdruck der erzeugten Maskendatei "kundens1.msk" | 102 |
| 71 | Teil des Ausdrucks der Textdatei "kundens1.txt" | 102 |
| 72 | Struktur der Datenbankdatei "aufpos" | 104 |
| 73 | Datenbankdatei "aufpos" | 104 |
| 74 | Befehlsfolge fuer gleichzeitiges Bearbeiten von Dateien | 106 |
| 75 | Datenbankdatei "aufpos" nach der Bearbeitung | 107 |
| 76 | Datenbankdatei "rech" | 108 |
| 77 | Aktualisierte, nach Auftragsnummern indizierte Auftragsdatei namens "auftrag" | 109 |
| 78 | Mittels JOIN gebildete Datenbankdatei "kundrech" | 111 |
| 79 | DBD-Auswahl-Menue des Viewgenerators | 113 |
| 80 | Relationen-Menue des Viewgenerators | 114 |
| 81 | Felder-auswaehlen-Menue des Viewgenerators | 114 |
| 82 | DISPLAY STATUS nach Aktivierung der Viewdatei "aufbesv" | 117 |
| 83 | Auflistung nach Aktivierung der Viewdatei "aufbesv" | 117 |

Seite

Bild 84 Aufruf und Abarbeiten von "progl"..... 121
85 Zusammenstellung wesentlicher Informationen ueber einen Kunden..... 124
86 "prog2" - Programm zur Erstellung eines Kundenbeleges..... 127
87 Programm "rechnung", mit Programmschleife realisiert..... 128
88 Programm "progkun1" zur Erstellung von Kundenbelegen mit Programmschleife realisiert... 130
89 Programm "progkun2" zur Erstellung von Kundenbelegen mit Programmverzweigung..... 133
90 Menueauswahl..... 134
91 Menueprogramm "menuel"..... 135
92 Beispiel eines Prozeduraufrufs..... 138
93 Beispiel einer Prozedurdatei..... 139
94 Programm "aufnahme"..... 140
95 Eroeffnung einer Katalogdatei mit SET CATALOG TO ? ..... 152
96 Struktur einer Katalogdatei..... 154
97 Eroeffnung einer Datenbankdatei mit USE ? ..... 157
98 Auswahl zur Datenbankdatei gehoeriger Indexdateien ..... 158
99 Tabelle eines Lagerbestandes ..... 167
100 Uebersicht zur Speichervariablen-Verwaltung .... 185
101 Beispiel zur Kellerung der Speichervariablen ... 189

## Einfuehrung

Die Effektivitaet des Einsatzes moderner Rechentechnik wird im wesentlichen durch die Qualitaet verfuegbarer Software bestimmt. Datenbankbetriebssysteme haben ihren festen Platz im Angebot an Standardsoftware. So bewaehrten sich die vom Kombinat ROBOTRON bereitgestellten Datenbankbetriebssysteme REDABAS fuer 8-Bit- oder 16-Bit-Mikrocomputer unter dem Betriebssystem SCP oder SCP1700 und REDABAS-3 fuer 16-Bit-Mikrocomputer unter DCP.

Aufbauend auf diesen Softwareprodukten mit grosser Anwendungsbreite wurde fuer die unter Steuerung des Betriebssystems DCP laufenden 16-Bit-Mikrocomputer das gegenueber REDABAS-3 in seinem Leistungsspektrum erweiterte Datenbankbetriebssystem REDABAS-4 entwickelt.

REDABAS-4 basiert auf dem relationalen Datenmodell. Die Daten werden tabellenorientiert verwaltet, so dass Datenstrukturen einfach und schnell definiert, benutzt und geaendert werden koennen. Hohe Flexibilitaet im Datenzugriff ist gewaehrleistet. Die nur einmal einzugebenden Daten stehen fuer unterschiedliche Einsatzmoeglichkeiten und Anwendungsprogramme zur Verfuegung. Die Datenbestaende lassen sich den wechselnden und wachsenden Informationsbeduerfnissen des Anwenders leicht anpassen. Dabei wird weitestgehend Datenunabhaengigkeit erreicht, d.h. bestehende Programme werden von Datenbankaenderungen nicht beeinflusst.
Ein besonderer Vorzug von REDABAS-4 ist die integrierte Befehlssprache, die in einem einheitlichen Sprachkonzept Datenverwaltung und Anwendungsprogrammierung gestattet.
REDABAS- oder REDABAS-3-Nutzer wissen diese guten Eigenschaften zu schaetzen und werden auch REDABAS-4 sofort problemlos handhaben koennen.
Die Benutzung von REDABAS-4 ist ausserdem so einfach gestaltet, dass auch Anwender ohne jegliche rechentechnischen Vorkenntnisse nach kurzer Einarbeitungszeit ihre Aufgaben loesen koennen. Eine Menuesteuerung (Befehl ASSIST) und ein jederzeit aufrufbares Hilfssystem (Befehl HELP) unterstuetzen sie dabei.
REDABAS-4 erlaubt sowohl die interaktive, operative Arbeitsweise als auch ein Abarbeiten umfangreicher, vorgefertigter Programme.

## 1. Vergleich der Leistungsparameter von REDABAS-4 mit Vorgaengersystemen

Als Vorgaengersysteme von REDABAS-4 sind REDABAS fuer 8-Bit-Mikrocomputer unter dem Betriebssystem SCP oder fuer 16-Bit-Mikrocomputer unter dem Betriebssystem SCP1700 und REDABAS-3 fuer 16-Bit-Mikrocomputer unter dem Betriebssystem DCP anzusehen. Sehr viele REDABAS-4-Anwender werden bereits Erfahrungen im Umgang mit wenigstens einem dieser Systeme gewonnen haben und mit dem Leistungsumfang vertraut sein. Vor ihnen wird die Frage stehen, wie sich die Systeme in ihrem Leistungsvermoegen und in ihrer Handhabung unterscheiden und ob man Dateien und Programme eines Vorgaengersystems in REDABAS-4 uebernehmen kann.

Generell baut REDABAS-4 auf seinen Vorgaengersystemen auf und uebernimmt bzw. erweitert deren gute Eigenschaften. Waehrend der Uebergang von REDABAS-3 zu REDABAS-4 mit Erhoehung des Nutzerservices in der gleichen Rechner- und Betriebssystemklasse nicht so erheblich ist, ist der Uebergang von REDABAS (schon zu REDABAS-3) zu REDABAS-4 durch volle Ausnutzung des 16-Bit-Mikrocomputers (Speicherkapazitaet und Geschwindigkeit) unter Steuerung des Betriebssystems DCP schwerwiegender. Die Handhabung von REDABAS-4 unterscheidet sich prinzipiell nicht von der der Vorgaengersysteme.

## 1.1. Vergleich quantitativer Leistungsparameter

Ein Vergleich der nachfolgend aufgefuehrten quantitativen Leistungsparameter von REDABAS-4 mit seinen Vorgaengersystemen zeugt von dem gewachsenen Potential von REDABAS-4 gegenueber REDABAS, waehrend Unterschiede zwischen REDABAS-4 und REDABAS-3 hier noch nicht sichtbar werden.

| | REDABAS-4<br>REDABAS-3 | REDABAS |
|---|---|---|
| max. Anzahl von Dateien | beliebig | beliebig |
| Anzahl der waehrend der Verarbeitung eroeffneten Dateien | max. 15 | max. 15 |
| Anzahl der waehrend der Verarbeitung eroeffneten Datenbankdateien | max. 10 | max. 2 |
| Saetze pro Datei | max. 1 Milliarde | max. 65535 |
| Zeichen pro Satz | max. 4000 | max. 1000 |
| Felder pro Satz | max. 128 | max. 32 |
| Feldtypen und Feldlaenge (in Byte) | alphanum. (254)<br>numerisch (19)<br>logisch (1)<br>Datum (8)<br>MERK (4 KB) | alphanum. (254)<br>numerisch (10)<br>logisch (1) |
| Sortierkriterien | max. 10 | max. 1 |
| Prozeduren je Prozedurdatei | max. 32 | - |
| Anzahl Speichervariablen | 256<br>(global + lokal) | 64 (global) |

## 1.2. Vergleich qualitativer Leistungsparameter

Der Vergleich qualitativer Leistungsparameter wird fuer Umsteiger von REDABAS auf REDABAS-4 in Abschnitt 1.2.1. und fuer Umsteiger von REDABAS-3 auf REDABAS-4 in Abschnitt 1.2.2. gefuehrt.

### 1.2.1. Vergleich REDABAS - REDABAS-4

REDABAS-4 erweitert betraechtlich das Leistungsspektrum von REDABAS, wobei ein Wechsel des Betriebssystems von SCP auf DCP und gegebenenfalls ein Wechsel von 8-Bit- auf den 16-Bit-Mikrocomputer vollzogen werden muss.
REDABAS ist zu REDABAS-4 bedingt kompatibel, d.h. Programme und Dateien von REDABAS (REDABAS-4) laufen nicht unveraendert unter REDABAS-4 (REDABAS).
Wenn jedoch das vom VEB Kombinat Robotron bereitgestellte Transformierprogramm RETRANS auf REDABAS-Dateien und - Programme angesetzt wird, laufen die in das REDABAS-3-Format (kompatibel zum REDABAS-4-Format) umgewandelten Dateien und Programme unter REDABAS-4. Die Transformation ist fuer alle in REDABAS erklaerten Dateitypen moeglich (vgl. hierzu Kap. 7: Umsteigen von REDABAS auf REDABAS-4).
REDABAS-4 zeichnet sich gegenueber REDABAS durch eine Reihe neuer bzw. erweiterter Befehle, Funktionen, Operationen, Daten- und Dateitypen aus.
So wurden Katalog-, Query-, Screen- und Viewdatei als Dateitypen (vgl. Absch. 8.1.3.) und "Datum" und "Merk" als Datentypen neu aufgenommen.

Fuer den Typ "Datum" stehen Datums- und Konvertierungsfunktionen zur Verfuegung, die installierte Datumsarithmetik erlaubt das Addieren und Subtrahieren von Werten sowie die Differenzbildung zweier Datumsangaben. Mit den Merkfeldern lassen sich jedem Datensatz variabel lange Textabschnitte zuordnen. Zusaetzlich zum Reportgenerator gibt es bei REDABAS-4 auch einen Etiketten-, Query-, Masken- und Viewgenerator.

Mit REDABAS-4 koennen von anderen Programmsystemen erstellte, im ASCII-Format vorliegende Dateien gelesen werden. Andere Programme koennen aufgerufen werden, ohne dass REDABAS-4 verlassen werden muss.

Fuer Neulinge steht zusaetzlich der ASSIST-Befehl zur Verfuegung, der ein menuegesteuertes Arbeiten gestattet. Der (schon von REDABAS) bekannte HELP-Befehl gibt wunschgemaess gezielte Auskunft ueber einzelne Befehle, Funktionen oder Sachverhalte. Die Moeglichkeiten der Korrektur, Cursorbewegung usw. werden bei den einzelnen Datenbankoperationen in Hilfstafeln angezeigt.
Die Erweiterung von Befehlen zeigt sich u.a. in einer differenzierteren Datensatzauswahl (vergl. Abschn. 9.3.2.) und in der Moeglichkeit des Anfuegens eines durch && eingeleiteten Kommentars.

### 1.2.2. Vergleich REDABAS-3 - REDABAS-4

REDABAS-4 erweitert das Leistungsspektrum von REDABAS-3 fuer die gleiche Rechnerklasse (16-Bit-Mikrocomputer) und das gleiche Betriebssystem (DCP). REDABAS-3 ist zu REDABAS-4 aufwaertskompatibel, d.h. alle Programme und Dateien von REDABAS-3 laufen unveraendert auch unter REDABAS-4, waehrend die Umkehrung im allgemeinen nicht gilt.

REDABAS-4 arbeitet mit 4 zusaetzlichen Dateitypen:
Katalog-, Query-, Screen- und Viewdatei (vgl. Abschn. 8.1.3.) und entsprechenden Befehlen fuer ihre Behandlung. (vgl. Kap. 6 und Abschnitte 3.9., 3.10. bzw. 3.11.4.).

Fuer eine Reihe von Befehlen wurde eine einheitliche Benutzeroberflaeche geschaffen, wozu der aus REDABAS-3 bekannte, aber jetzt erweiterte Report- bzw. Etikettengenerator und ein Query-, Masken- und Viewgenerator gehoeren.
Einige Befehle wurden erweitert, z.B. um die Moeglichkeit der gleichzeitigen Angabe der FOR- und WHILE-Klausel in einem Befehl, der bisher nur eine alternative Verwendung gestattete, oder das Einfuegen eines Kommentars am Ende einer Befehlszeile, der mit den Zeichen && eingeleitet wird. Neue Befehle und Funktionen sind zur Erhoehung der Leistungsfaehigkeit hinzugekommen. Auf diese Weise werden unter anderen die Einbindung von Assemblerroutinen in REDABAS-4-Programme, die Verbesserung der Fehlerkorrektur und Testung von Programmen und die Beruecksichtigung der Systemumgebung ermoeglicht. Besonders sei auf die Moeglichkeit des Holens von Befehlen aus dem HISTORY-Speicher hingewiesen (vgl. Befehl SET HISTORY in Abschn. 9.3.3.).

## 2. Arbeitsweise

### 2.1. Voraussetzungen fuer die Anwendung

REDABAS-4 arbeitet unter dem Betriebssystem DCP oder einem kompatiblen Betriebssystem.

Hardwaremaessig wird ein 16-Bit-Personalcomputer mit mindestens 384 KBytes RAM und zwei Diskettenlaufwerken oder einem Diskettenlaufwerk und einem Festplattenlaufwerk gefordert.

REDABAS-4 stellt fuer die Datenbankverwaltung und Programmierung eine eigene Datenmanipulationssprache zur Verfuegung, in der die Anweisungen zur Bearbeitung der Datenbank formuliert werden.

Bevor Sie mit REDABAS-4 arbeiten, muessen Sie sich mit dem Befehlsvorrat von REDABAS-4 vertraut machen. Im Kapitel 3 werden die Funktionen der Datenbankverwaltung vorgestellt, und an einem Demonstrationsbeispiel wird die Anwendung sofort erlaeutert. Fuer das Vertrautsein mit REDABAS-4 und eine rasche Praxiswirksamkeit ist zu empfehlen, dieses Demonstrationsbeispiel auf dem eigenen Personalcomputer nachzuvollziehen.

Es ist auch moeglich, Befehle zu Programmen zusammenzufassen, zu speichern und wunschgemaess aufzurufen. Die Programmierung mit REDABAS-4 wird im Kapitel 4 vorgestellt.
Eine detaillierte Beschreibung aller REDABAS-4-Befehle und -Funktionen finden Sie im Kapitel 9.

## 2.2. Allgemeine Hinweise

Bevor Sie mit der eigentlichen Handhabung von REDABAS-4 beginnen, moechten wir Sie mit einigen allgemeinen Hinweisen und den in dieser Programmtechnischen Beschreibung verwendeten Regeln vertraut machen.

- Grossgeschriebene Ausschriften in den Bildschirmbeispielen weisen auf einen REDABAS-4-Befehl.

- Variable Angaben in REDABAS-4-Befehlen (z. B. Datei- und Feldnamen) sind klein geschrieben.

- "..." wird verwendet, um bestimmte Eingaben (Befehle, Mitteilungen, Feld- oder Dateinamen,...) im Text dieser Beschreibung hervorzuheben. Diese Anfuehrungszeichen duerfen Sie nicht eingeben! Innerhalb der als Beispiel angegebenen Befehlszeile sind die Anfuehrungszeichen jedoch Begrenzer von Zeichenreihenkonstanten und muessen mit eingegeben werden.

- <ET> bedeutet, dass Sie die <ET>-Taste, (die auf mancher Rechnertastatur auch als RETURN, <--', ETl oder ENTER bezeichnet ist) druecken sollen. Geben Sie nicht die Buchstaben ET und auch nicht die Klammern ein!
  Das Gleiche gilt fuer die <CTRL>-Taste!
  Bitte beachten Sie ausserdem, dass die folgenden, in Bildschirmausschriften vorkommenden Symbole den in dieser Schrift verwendeten Notationen entsprechen:
  ^ entspricht <CTRL>
  <--' entspricht <ET>.

- Fuer die Notation von REDABAS-4-Befehlen gilt:

  [...] Schrift in eckigen Klammern ist Teil eines REDABAS-4-Befehls, der wahlweise angegeben werden kann.
  <...> Kleingeschriebenes in spitzen Klammern ist Teil eines REDABAS-4-Befehls, der mit echter Information ausgefuellt werden muss.
  / Der Schraegstrich weist auf eine alternative Benutzung der Klausel.

  Beispiel: CREATE <db-datei>
  ---------
  Sie muessen anstelle <db-datei> den Namen einer zu erzeugenden Datenbankdatei eingeben, z.B. kunden.

  Vergleichen Sie hierzu Abschnitt 9.1.

- Grundregeln der REDABAS-4-Befehlssprache:
  Es ist im allgemeinen gleichgueltig, ob Befehle in Gross- oder Kleinbuchstaben geschrieben werden. Wegen der besseren Unterscheidung und Lesbarkeit von Programmen wird jedoch empfohlen, Befehle gross und Datei- und Feldnamen klein zu

schreiben.

Ein Befehl darf max. 254 Zeichen lang sein.

In einer Befehlszeile muss das erste, vom Leerzeichen verschiedene Zeichen zum Befehlswort selbst gehoeren. Die Zusatzklauseln koennen in beliebiger Reihenfolge angehaengt werden. Ein Befehlswort ist z. B. "CREATE" oder "INSERT". Eine Zusatzklausel spezifiziert die durch ein Befehlswort eingeleitete Aktion, z. B. wird mit "FOR" eine Zusatzklausel eingeleitet.

Die Gesamtheit von Befehlswoertern und Einleitungswoertern von Zusatzklauseln bilden die sogenannten "Steuerwoerter" von REDABAS-4. Sie sind in der Programmtechnischen Beschreibung immer in Grossbuchstaben geschrieben.

Steuerwoerter werden durch ihre ersten vier Buchstaben eindeutig identifiziert, d. h. Steuerwoerter duerfen auf vier Buchstaben abgekuerzt werden. Beispielsweise kann DISPLAY STRUCTURE auf DISP STRU verkuerzt werden.
Nach der Eingabe eines Befehles ist die <ET>-Taste zu betaetigen.

Damit Sie REDABAS-4 schon nach kurzer Einarbeitungszeit nutzen koennen, werden in den Kapiteln 3 und 4 dieser Programmtechnischen Beschreibung die REDABAS-4-Befehle und -Funktionen nach Moeglichkeit in der Reihenfolge und Notation vorgestellt (und an Beispielen demonstriert), in der Sie sie fuer die praktische Arbeit brauchen. Bewusst wurde dabei auf das vollstaendige Befehlsspektrum und auch auf die vollstaendige Beschreibung aller Moeglichkeiten eines Befehls verzichtet. Im Abschnitt 9.3. finden Sie eine lueckenlose Darstellung saemtlicher Befehle mit all ihren Varianten in alphabetischer Reihenfolge geordnet, Abschnitt 9.2. erklaert ausfuehrlich alle REDABAS-4-Funktionen.

## 2.3. Inbetriebnahme von REDABAS-4

### 2.3.1. Was Sie vor dem ersten Start von REDABAS-4 tun muessen

Das Inbetriebnehmen von REDABAS-4 setzt keinen gesonderten Installationsvorgang voraus. Allerdings sind vor dem ersten Starten von REDABAS-4 einige e i n m a l i g noetige Vorbereitungen zu treffen:

#### Das Anfertigen von Sicherheitskopien

Fertigen Sie unbedingt ein bis zwei Duplikate von Ihrer REDABAS-4-Diskette an und benutzen Sie diese Duplikate zur Arbeit. Damit wird verhindert, dass bei einer unbeabsichtigten Zerstoerung des Disketteninhaltes (z. B. Geraete- oder Bedienfehler) die Originalkomponenten verlorengehen.

Vor dem Kopieren versehen Sie Ihre Originaldiskette mit einem Schreibschutz-Aufkleber. Zum Kopieren benutzen Sie das DCP-Kommando DISKCOPY. Anschliessend koennen Sie sich mittels DISKCOMP davon ueberzeugen, dass Ihr Duplikat mit dem Original uebereinstimmt. Auch mittels des DCP-Kommandos COPY koennen Sie alle Bestandteile der REDABAS-4-Diskette kopieren (Kontrolle hier durch COMP). Informieren Sie sich im DCP-Handbuch, falls Sie diese und die spaeter erwaehnten DCP-Kommandos nicht kennen!

## Das Konfigurieren des DCP

Damit das Betriebssystem DCP Ihre Datenbankanwendung effektiv unterstuetzen kann, muessen die beiden Systemparameter FILES und BUFFERS gegenueber den sonst im DCP geltenden Standardwerten erhoeht werden. Dazu dient eine Datei mit dem Namen CONFIG.SYS. Sie muss sich neben anderen DCP-Komponenten beim Starten des Betriebssystems im Stammverzeichnis desjenigen Datentraegers (Diskette oder Festplatte) befinden, von dem aus das Betriebssystem gestartet werden soll.
Ist die CONFIG.SYS-Datei dort bereits vorhanden, ueberpruefen Sie deren Inhalt (z.B. mit dem DCP-Kommando TYPE). Ergaenzen Sie

```
FILES=20
BUFFERS=15
```

bzw. erhoehen Sie die bereits dafuer vorgesehenen Werte, falls sie nicht schon groesser oder gleich diesen fuer REDABAS-4 guenstigen Angaben sind. (Zum Aendern von CONFIG.SYS benutzen Sie EDLIN oder einen anderen verfuegbaren Editor, zur Neueingabe der Datei ist auch COPY CON: CONFIG.SYS moeglich.)

Sie koennen auch die mit REDABAS-4 gelieferte CONFIG.SYS-Datei benutzen und erforderlichenfalls durch Editieren ergaenzen. Beachten Sie aber, dass diese Datei nur dann wirksam wird, wenn sie in das Stammverzeichnis des Datentraegers kopiert wird, von dem aus das Betriebssystem gestartet wird.
Ueberhaupt beruecksichtigt das DCP Ihre in CONFIG.SYS vorgenommenen Aenderungen erst bei einem erneuten Betriebssystem-Start, und nur dann, wenn CONFIG.SYS im entsprechenden Stammverzeichnis vorgefunden wird.
Das Bereitstellen und/oder Anpassen von CONFIG.SYS ist ein einmaliger Vorgang zum Einrichten Ihres Betriebssystems fuer die kuenftige Anwendung von REDABAS-4.

## Die Uebernahme von REDABAS-4 auf die Festplatte

Wenn Ihr Computer eine Festplatte besitzt, ist es auf jeden Fall zweckmaessig, REDABAS-4 auf der Festplatte zu installieren und von dort aus zu starten. Im allgemeinen ist es ausreichend, wenn auf der Platte 2 bis 2 1/2 Megabytes Speicherplatz fuer Programme und Dateien verfuegbar sind.

Folgende Schritte sind erforderlich:

- Einrichten eines Unterverzeichnisses z. B. mit dem Namen REDABAS fuer die REDABAS-4-Systemprogramme

  ```
  MKDIR C:\REDABAS
  ```

- Einstellen auf dieses Unterverzeichnis, d. h. Erklaeren dieses Unterverzeichnisses zum Standardverzeichnis (aktuellen Verzeichnis) im Laufwerk C mit dem DCP-Kommando

  ```
  CHDIR C:\REDABAS
  ```

- Einlegen der REDABAS-4-Systemdiskette in das Laufwerk A und Kopieren aller REDABAS-4-Systemprogramme in das Unterverzeichnis, z. B.
  COPY A:REDA*.* C:/V
- Entnehmen der nun nicht mehr benoetigten REDABAS-4-Diskette aus dem Laufwerk A.

### 2.3.2. Der REDABAS-4-Start

Unter Beachtung der im vorigen Abschnitt beschriebenen Vorarbeiten starten Sie REDABAS-4 auf folgende Weise:

1) Schalten Sie den Computer ein und starten Sie DCP mit der angepassten Datei CONFIG.SYS!
   Geben Sie das aktuelle Datum und die Uhrzeit ein!
   Beachten Sie, dass REDABAS-4 Datum und Uhrzeit anzeigen, auswerten und in Felder von Datenbankdateien oder generierte Listen uebernehmen kann. Deshalb ist die Eingabe der aktuellen Werte (mit den DCP-Kommandos DATE und TIME) bei jedem Betriebssystemstart, spaetestens aber vor dem Aufruf von REDABAS-4 erforderlich.

2) Die weitere Vorgehensweise ist von der Konfiguration Ihres Computers abhaengig.
   Ist Ihr Computer m i t e i n e r F e s t p l a t t e ausgeruestet, sind folgende Handlungen auszufuehren:
   . Erklaeren Sie die Festplatte zum Standardlaufwerk!
   . Stellen Sie das Unterverzeichnis, das die REDABAS-4-Systemprogramme enthaelt, als aktuelles Verzeichnis ein!

3) Haben Sie k e i n e F e s t p l a t t e , starten Sie REDABAS-4 von Diskette (Laufwerk A oder B). Dabei ist zu unterscheiden, ob sich die REDABAS-4-Systemprogramme auf einer Betriebssytemdiskette befinden oder nicht.
   Soll auf eine Betriebssystemkette von 720 KBytes REDABAS-4 uebertragen werden, so ist auf die Betriebssystem-Bestandteile zu verzichten, die Sie nicht gemeinsam mit REDABAS-4 benutzen wollen.
   Andererseits sind die verborgenen Programme des DCP-Kerns, der Kommandoprozessor COMMAND.COM und die beim DCP-Start automatisch aufgerufenen Bestandteile unbedingt erforderlich (ebenso solche Programme, die Sie von REDABAS-4 aus durch den RUN-Befehl (vgl. Abschn. 5.1.) aufrufen wollen oder die den REDABAS-4-internen Texteditor ersetzen sollen (vgl. Abschn. 10.1.3.)).
   Befinden sich die REDABAS-4-Systemprogramme nicht auf der Betriebssystem-Diskette, dann legen Sie die REDABAS-4-Diskette in das freie Laufwerk und erklaeren dieses zum aktuellen Laufwerk.
   Andernfalls ist nur dann eine Aktion erforderlich, wenn die REDABAS-4-Systemprogramme nicht gemeinsam mit dem Betriebssystem im Stammverzeichnis der Diskette stehen, sondern in einem speziellen Unterverzeichnis. Dieses Unterverzeichnis ist als aktuelles Verzeichnis einzustellen.

4) Das Betriebssystem meldet sich mit dem Bereitschaftszeichen. Sie tippen ein

   REDABAS

   und druecken anschliessend die <ET>-Taste.

5) REDABAS-4 meldet sich mit dem Startbild. Sie koennen Ihre Arbeit mit REDABAS-4 beginnen.
   Das Bereitschaftszeichen von REDABAS-4 besteht aus Doppelpunkt und Leerzeichen. Das heisst, immer wenn Sie am Anfang einer Bildschirmzeile einen Doppelpunkt und dahinter den Cursor sehen, koennen Sie einen REDABAS-4-Befehl eingeben.
   Sie koennen aber auch die Taste <F1> zur Aktivierung des HELP-Befehls oder die Taste <F2> zum Aufruf des ASSIST-Befehls betaetigen. Beide geben Ihnen Hilfestellung bei der REDABAS-4-Arbeit. Im Abschnitt 2.4. bzw. 9.3.3. werden beide Befehle ausfuehrlich besprochen.

Zur Erleichterung und Beschleunigung der Eingabe von REDABAS-4-Befehlen moechten wir Sie auf folgende Moeglichkeiten hinweisen:

- Wie der Gebrauch der Funktionstasten <F1> und <F2> zum Aufruf von Befehlen bereits beschrieben wurde, kann den Tasten <F2> bis <F10> ein beliebiger REDABAS-4-Befehl ueber den SET FUNCTION-Befehl (vgl. Abschn. 9.3.3.) oder die Konfigurierungsdatei REDAB.CF (vgl. Abschn. 10.1.2.) zugewiesen werden. Anstatt den Befehl durch die ihn bildende Buchstabenfolge einzutippen, brauchen Sie nur die entsprechende Funktionstaste zu druecken. Wenn in Ihrem System noch keine diesbezueglichen Aenderungen durchgefuehrt wurden, erhalten Sie mit der Taste <F6> (DISPLAY STATUS) u.a. die Zuordnung der programmierbaren Funktionstasten zu voreingestellten Befehlen.

- Sie koennen die Dienste des HISTORY-Puffers nutzen, um bereits einmal eingegebene Befehle zu wiederholen bzw. zu korrigieren. Wenn Sie die voreingestellten Werte ueber SET HISTORY (vgl. Abschnitt 9.3.3.) oder die Konfigurierungsdatei REDAB.CF (vgl. Abschn. 10.1.2.) nicht veraendert haben, koennen Sie auf die 20 zuletzt eingegebenen REDABAS-4-Befehle zugreifen.
  Durch einmalige bzw. mehrmalige Betaetigung der Aufwaerts-Pfeiltaste holen Sie den letzten bzw. die weiter vorher eingegebenen Befehle aus dem HISTORY-Puffer. Mit der Abwaerts-Pfeiltaste bewegen Sie sich wieder in Richtung des letzten Befehls zurueck. Wenn Ihr gesuchter Befehl angezeigt wird, ist er mit <ET> absendbar oder mit Hilfe folgender Tasten korrigierbar, die ebenfalls bei der normalen Befehlseingabe verwendet werden koennen:

| | |
|---|---|
| Rechts-Pfeiltaste | bewegt Cursor nach rechts |
| Links-Pfeiltaste | bewegt Cursor nach links |
| <CTRL-F> | bewegt Cursor ein Wort nach rechts |
| <CTRL-A> | bewegt Cursor ein Wort nach links |
| <CTRL-T> | loescht Wort rechts vom Cursor |
| Ruecktaste | loescht Zeichen links vom Cursor |
| <Del> oder <CTRL-G> | loescht Zeichen an der Cursorposition |
| <Ins> oder <CTRL-V> | Einfuegen EIN/AUS. Bei EIN werden neue Zeichen eingefuegt, vorhandene nach rechts geschoben. Bei AUS werden vorhandene Zeichen ueberschrieben. |

- Waehrend REDABAS-4 mit irgendeinem Vorgang beschaeftigt ist, duerfen Sie als routinierter Nutzer bereits ueber die Tastatur die naechste Eingabe vollziehen. Diese Eingabe wird in einem Zwischenspeicher, dem sogenannten TYPE-AHEAD-Puffer, aufbewahrt und von REDABAS-4 abgefordert. Die Standardkapazitaet des TYPE-AHEAD-Puffers betraegt 20 Zeichen. Die Kapazitaet kann durch den SET TYPEAHEAD-Befehl (vgl. Abschn. 9.3.3.) oder die Konfigurierungssdatei REDAB.CF (vgl. Abschn. 10.1.2.) veraendert werden.

Wir moechten Sie ausserdem an dieser Stelle mit einem Befehl bekanntmachen, mit dem Sie zu Beginn oder auch waehrend einer Sitzung mit REDABAS-4 bestimmen koennen, welches Laufwerk Sie zum Standardlaufwerk fuer die Dateneingabe und -ausgabe waehlen.
(Zunaechst gilt das Laufwerk, das beim Start von REDABAS-4 im DCP als Standardlaufwerk eingestellt war (am Bereitschaftszeichen des DCP zu erkennen) auch in REDABAS-4 als Standardlaufwerk.)
Mit dem REDABAS-4-Befehl

SET DEFAULT TO <laufwerk> <ET>

koennen Sie davon abweichend veranlassen, dass alle Dateien und Programme, die Sie erstellen, automatisch auf den Datentraeger in dem spezifizierten <laufwerk> geschrieben bzw. von dort gelesen werden. Fuer die REDABAS-4-Programmkomponenten selbst bleibt das beim Start benutzte Laufwerk aktuell.

Ein fortgeschrittener REDABAS-4-Anwender hat die Moeglichkeit, Standardparameter und Schalterstellungen von REDABAS-4 bequem zu modifizieren, indem er eine Konfigurierungsdatei (REDAB.CF) bereitstellt, auf die bei jedem REDABAS-4-Start zurueckgegriffen wird. Interessierte Anwender finden eine detaillierte Beschreibung im Abschnitt 10.1. dieser Programmtechnischen Beschreibung.

### 2.3.3. Das Beenden

Das Beenden von REDABAS-4 geschieht immer mit dem Befehl

QUIT

QUIT veranlasst, dass a l l e Dateien geschlossen werden, die Daten auf Diskette oder Festplatte gesichert werden, die Ausfuehrung von REDABAS-4 beendet wird und die Steuerung an das

Betriebssystem uebergeben wird.

Achtung! Sie duerfen niemals REDABAS-4 verlassen, ohne den Befehl QUIT gegeben zu haben. Jeder Programmabbruch (z. B. Ausschalten des Rechners) kann zu Datenverlusten fuehren.
Nach der Eingabe von QUIT und dem Druecken der <ET>-Taste erscheint die REDABAS-4-Endenachricht. Anschliessend meldet sich das Betriebssystem wieder.
Sie koennen nach der Bereitschaftsmeldung des Betriebssystems REDABAS-4 beliebig oft starten (auch wenn Sie zwischendurch mit anderen Systemen gearbeitet haben). Ein erneutes Starten des DCP ist dann nicht erforderlich.

## 2.4. Die Hilfsmoeglichkeiten ASSIST und HELP

REDABAS-4 bietet drei Ebenen einer Hilfestellung:

- Fuer den Erstanwender von REDABAS-4 ohne jegliche Erfahrung den "Assistenten". Durch Eingabe von

  ASSIST oder Druecken der <F2>-Taste

  wird REDABAS-4 in ein menuegesteuertes Datenbankbetriebssystem verwandelt. Wichtige Datenbankoperationen werden in Menueform vorgestellt, kommentiert und aufgerufen.
  ASSIST wird vor allem fuer das Erlernen von REDABAS-4 angeboten.

- Der erfahrenere Anwender wird nicht mehr mit den ASSIST-Menues arbeiten. Er verwendet die Befehle und ruft - beispielsweise wenn er sich ueber die Syntax eines Befehls im Zweifel ist - den Befehl

  HELP

  auf.
  Aehnlich wie in der Programmtechnischen Beschreibung werden Erlaeuterungen zu Befehlen und Funktionen oder allgemeine Erklaerungen zu REDABAS-4 geboten.
  HELP kann jederzeit durch Betaetigen der <F1>-Taste aktiviert werden.

- Unkorrekte Eingaben werden generell durch Fehleranzeigen dokumentiert.

ASSIST und HELP erfuellen unterschiedliche Zwecke. Mit den ASSIST-Menues werden nur Neulinge arbeiten, mit HELP kann sich auch ein erfahrener Anwender bei Bedarf eine Auskunft holen. Generell ist das Arbeiten ohne den ASSIST-Modus effektiver.

Wir moechten Ihnen die Handhabung von ASSIST an einem trivialen Beispiel, naemlich dem Anzeigen des Dateiverzeichnisses Ihrer REDABAS-4-Diskette, erlaeutern. Wir empfehlen Ihnen, das Beispiel auf Ihrem Rechner zu probieren. Lassen Sie sich nicht durch die auf Sie einstuermende Begriffsvielfalt verwirren!
Sie eignen sich das notwendige datenbankspezifische Wissen beim Studieren der Kapitel 3 und 4 leicht an. Eine ausfuehrliche Beschreibung des ASSIST-Befehls ist in Abschnitt 9.3.3. zu finden.

Wir setzen voraus, dass Sie REDABAS-4 von einem Duplikat Ihrer REDABAS-4-Diskette im Laufwerk A gestartet haben. (Falls REDABAS-4 schon auf der Festplatte installiert wurde, ist die Beschreibung mit einem REDABAS-4-Start von der Festplatte mit veraenderten Laufwerkangaben zutreffend.)

Nach Eingabe von ASSIST mit <ET> und anschliessendem Druecken der Links-Pfeiltaste sind Sie von dem voreingestellten Komplex Auswahl (nach links) auf den in der Zeile zuletzt angeordneten Komplex Dienste gesprungen.
(Sie haetten dafuer auch 7 mal die Rechts-Pfeiltaste druecken oder "d" als Anfangsbuchstabe des auszuwaehlenden Komplexes eingeben koennen.)

Auf Ihrem Bildschirm muss die 1. Zeile und das rechte Fenster aus Bild 1 zu sehen sein, wobei ganz rechts die aktuelle Uhrzeit angezeigt wird. Aus Darstellungsgruenden wurden in Bild 1 einige Begriffe durch Punkte abgekuerzt und einige Steuertasten durch druckbare Zeichen ersetzt. Auch alle weiteren Bilder stellen keine exakten Abbilder des Bildschirms dar.

```
-------------------------------------------------------------
|                                                           |
|Ausw. Neu Modus Posit. Extr. Organ. Aendern Dienste 15:36:12 |
|                                                           |
|                                                           |
|                                   |----------------------| |
|                                   |                      | |
| |-----------------------| |----| | Laufwerk setzen       | |
| | .DBD Datenbankdateien | | A: | | Datei kopieren        | |
| | .IDX Indexdateien     | | B: | | Dateiverzeichnis      | |
| | .MSK Maskendateien    | | C: | | Dateiname aendern     | |
| | .ETI Etikettendateien | | D: | | Datei loeschen        | |
| | .DEF Reportdateien    | | E: | | DBD-Struktur listen   | |
| | .TXT Textdateien      | | F: | |----------------------| |
| | .VUE Viewdateien      | |----| | Import von PFS-Format | |
| | .QRY Querydateien     |        | Export zu PFS-Format  | |
| | .SCR Screendateien    |        |----------------------| |
| | .*   Alle Dateien     |                                |
| |-----------------------|                                |
|                                                           |
|Befehl: DIR A:                                             |
|ASSIST     |<A:>|            |Opt: 1/10      |       |     |
|           Markierungsbalken - ^v. Auswahl - <--'.          |
|                   Waehlen Sie den Dateityp.                |
|                                                           |
-------------------------------------------------------------
```

Bild 1 Die Arbeit mit dem ASSIST-Befehl

Durch zweimalige Betaetigung der Abwaerts-Pfeiltaste bewegen Sie den Makierungsbalken (aktuelle, invers dargestellte Menueoption) auf die Option Dateiverzeichnis. Zu diesem Zeitpunkt koennen Sie sich durch Druecken der <F1>-Taste (entspricht dem Aufruf von HELP) ueber die Wirkung der aktuellen Menueoption und die Syntax des ihr zugrunde liegende Befehls informieren (vgl. Bild 2).

```
------------------------------------------------------------------
|                                                                |
|                          DIR                                   |
|                                                                |
| DIR zeigt das Inhaltsverzeichnis des selekt. Laufwerks an.     |
|                                                                |
|   Befehlsform:  DIR [<laufwerk>][<pfad\][<muster>]             |
|                                                                |
------------------------------------------------------------------
```

Bild 2 HELP-Information im ASSIST-Modus

Ein erneutes Druecken der <F1>-Taste laesst die HELP-Information verschwinden und das urspruengliche Bild erscheinen. Betaetigen Sie jetzt <ET>, wird zunaechst das Untermenue zur Laufwerkwahl angezeigt (vgl. Bild 1), wobei der Markierungsbalken auf "A:" steht. Nochmaliges <ET> fuehrt zum Untermenue zur Dateitypwahl. Dieser Zustand ist in Bild 1 festgehalten.

Auf dem Bildschirm wird allerdings das Untermenue zur Laufwerkwahl nicht mehr angezeigt. Mit <PgDn> springen Sie zur Option ".* Alle Dateien". Ein letztes <ET> loest schliesslich die gewuenschte Anzeige des Dateiverzeichnisses aus. Mit <Esc> verlassen Sie den ASSIST-Modus, dagegen das REDABAS-4-System mit der Option "Ende-REDABAS-4" aus dem Komplex Auswahl.

## 2.5. Betriebsarten und Aufteilung des Bildschirms

Bei der Arbeit mit REDABAS-4 sind mehrere Betriebsarten zu unterscheiden. Wir sprechen vom interaktiven Modus (oder auch von der Befehlsebene), wenn Sie nach der Anzeige des Bereitschaftszeichens Befehl um Befehl ueber die Tastatur eingeben. Es ist aber auch moeglich, Befehle zur Erfuellung einer Aufgabe in einer Programmdatei zusammenzustellen und die Programmdatei in der festgelegten Reihenfolge abzuarbeiten (vgl. Kap. 4 ). Es liegt dann der Programmodus vor. Genauer schaltet der Aufruf einer Programmdatei mit DO vom interaktiven Modus in den Programmodus um, und der RETURN-Befehl in der Programmdatei schaltet zurueck.

Der Programmodus bildet kein Hindernis, in eine Programmdatei Befehle aufzunehmen, die einen Eingriff des Benutzers erfordern, also eine interaktive Arbeitsweise darstellen. Es ist jedoch nicht moeglich, von der in der Programmdatei gespeicherten Befehlsfolge abzuweichen. Dagegen sind programmtypische Konstruktionen, wie z.B. Programmschleifen, im interaktiven Modus nicht erlaubt.

Eine spezielle Betriebsart ist der ASSIST-Modus, der nach Aufruf des ASSIST-Befehls anliegt. Als eine weitere spezielle Betriebsart gibt es den Seitenmodus (s. Abschn. 8.4.2.), in dem der Bildschirm gezielt adressiert werden kann.

Eine bestimmte Aufteilung des Bildschirms und eine feststehende Zuordnung von Steuertasten zu Funktionen, was auch als einheitliche Benutzeroberflaeche bezeichnet wird, gilt fuer Befehle, die mit dem Menuesystem arbeiten. Das betrifft die Befehle:

ASSIST, BROWSE, CREATE, CREATE/MODIFY LABEL, CREATE/MODIFY

QUERY, CREATE/MODIFY REPORT, CREATE/MODIFY SCREEN, CREATE/MODIFY VIEW, MODIFY STRUCTURE und SET.

Diese Befehle heissen menuegesteuerte Befehle, weil ihr Ablauf (bei BROWSE, CREATE und MODIFY STRUCTURE unter Zuhilfenahme von <CTRL-Home>) von der nachfolgend erklaerten Menuezeile aus ueber eventuell weitere Menueebenen gesteuert wird. Sie sind eine Teilmenge der seitenorientierten Befehle, und damit sind fuer sie die Cursortasten bzw. Tastenkombinationen im Seitenmodus massgebend (s. Abschn. 8.4.2.).
Im erweiterten Sinne gehoert auch HELP zu den menuegesteuerten Befehlen, jedoch weicht der Befehl von der einheitlichen Benutzeroberflaeche ab.
Von den die Zeilen 1 bis 25 umfassenden Bildschirm gilt fuer menuegesteuerte Befehle folgende Aufteilung des Bildschirms (vgl. Bild 3):

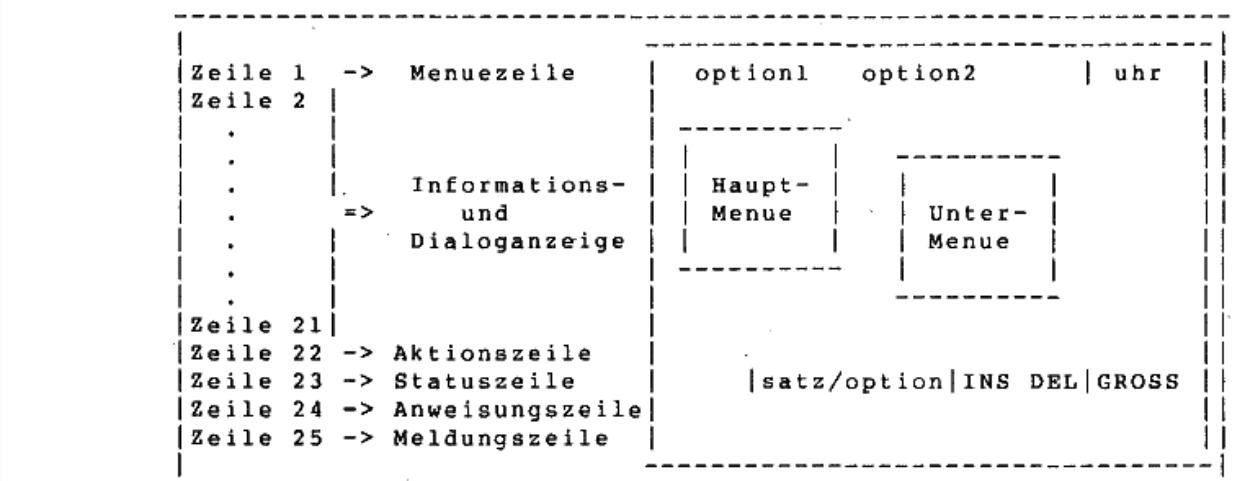



Bild 3 Aufteilung des Bildschirms

Zeile 1 ist der Menuezeile vorbehalten, die einzelne Optionen des Hauptmenues und eine Uhrzeitangabe enthaelt (vgl. Bild 1). Diese Optionen koennen wiederum in Form von Menues, sogenannten Pull-down-Menues, ausgelegt sein.

Die Zeilen 2 bis 21 dienen zur Informations- und Dialoganzeige. Die Informationsanzeige bietet Resultate des angeforderten Befehls oder Hilfestellung fuer die aktuelle Operation. Sie wird teilweise durch Betaetigung der <F1>-Taste auf dem Bildschirm an- bzw. abgestellt. Sie enthaelt z.B. das Hilfsmenue zur Cursorsteuerung oder im ASSIST-Modus HELP-Informationen (vgl. Bild 2).

Die Dialoganzeige weist Informationen auf, die durch Eingaben des Benutzers auszuwaehlen bzw. zu aktualisieren sind. So gehoeren z.B. die in einem Fenster angeordneten Pull-down-Menues, von denen weiterfuehrende Untermenues oder beim BROWSE-Befehl der Inhalt einer Gruppe von Datensaetzen zur Dialoganzeige.

Zeile 22 nimmt die Aktionszeile auf. Im ASSIST-Modus wird in dieser Zeile die syntaktische Form des ausgewaehlten Befehls aufgebaut (vgl. viertletzte Zeile in Bild 1). Andere menuege-

steuerte Befehle, wie z.B. CREATE/MODIFY VIEW, benutzen diese Zeile fuer befehlsspezifische Angaben.

Zeile 23 enthaelt die Statuszeile (vgl. drittletzte Zeile in Bild 1). Sie liefert ueber die aktuelle Arbeitsumgebung folgende Angaben:

Aufgerufener Befehl (ASSIST), aktuelles Laufwerk (A:), aktuelle Datei (frei), aktuelle Option- (1), Feld- oder Satznummer der jeweils insgesamt moeglichen Nummern (10), Anzeige der gedrueckten <Ins>- Taste und der Loeschmarkierung eines Satzes (beides frei) und zuletzt Anzeige der gedrueckten <Num Lock>- und <Caps Lock>- Taste (beides frei).

Zeile 24 ist als Anweisungszeile vorgesehen (vgl. vorletzte Zeile in Bild 1). Sie gibt Anweisungen an, wie man sich ueber den Bildschirm bewegt, eine Option auswaehlt oder die Arbeit fortsetzt. Im Fehlerfall wird in der Anweisungszeile eine Fehlermeldung ausgewiesen.

Zeile 25 wird Meldungszeile genannt (vgl. letzte Zeile in Bild 1). In ihr wird die eingestellte Menueoption erlaeutert.

Es sei noch auf folgenden Zusammenhang bezueglich der Statuszeile hingewiesen. Wenn Sie REDABAS-4 gestartet haben, erscheint das Bereitschaftszeichen auf Zeile 25. Es wird keine Statuszeile angezeigt. Gedrueckte <INS>-, <Num Lock>- und <Caps Lock>- Tasten werden in Zeile 1 protokolliert.
Das liegt daran, dass der fuer die Anzeige der Statuszeile verantwortliche SET STATUS-PARAMETER auf OFF und der fuer die Anzeige von REDABAS-4-Meldungen verantwortlichen SET SCOREBOARD-Parameter auf ON voreingestellt sind. Sie koennen auch mit angezeigter Statuszeile auf Zeile 23 arbeiten, in dem Sie den Befehl SET STATUS ON eingeben. Das Bereitschaftszeichen liegt dann auf Zeile 22 an und auf Zeile 25 wird eine Aufforderung zur Eingabe eines REDABAS-4-Befehls ausgewiesen.
Befehle, die im Seitenmodus arbeiten, halten waehrend dieser Betriebsart den SET STATUS-Parameter automatisch auf ON.

# 3. Funktionen der Datenbankverwaltung

## 3.1. Einrichten und Aendern von Datenbankdateien

### 3.1.1. Erstellen einer Datenbankdatei

#### Definieren der Struktur einer Datenbankdatei

Eine effektive Datenbankverwaltung erfordert Kenntnis des Informationsbedarfs der einzelnen Sachgebiete. Entsprechend erfolgt der logische Datenbankentwurf, d.h. konkret das Festlegen der Struktur der Dateien.

Beim Erstellen einer Datenbankdatei wird also gewissermassen der aeussere Rahmen festgelegt, in den die Daten aufgenommen werden sollen. Geht man von der Sicht des Anwenders aus, so sieht er seine als Datei abzuspeichernden Datenbestaende in Form einer Tabelle, deren Zeilen Datensaetzen und deren Spalten Datenfeldern entsprechen. Im Beispiel der Kundendatei ist eine komplette Information ueber den Kunden (Name, Vorname usw.), also eine Zeile, als Datensatz und die Menge der gleichartigen

Datenelemente (z.B. alle Namen), als eine Spalte, als Datenfeld realisiert.

Alle Datensaetze einer Datei (Zeilen) sind gleich strukturiert, d.h. dass nur die Struktur eines Datensatzes beschrieben zu werden braucht.

Im folgenden moechten wir Ihnen anhand eines Beispiels die mit REDABAS-4 ausfuehrbaren Funktionen der Datenbankverwaltung - beginnend mit dem Erstellen einer Datenbankdatei - demonstrieren.

- Ist REDABAS-4 bereit? (Es muss ein Doppelpunkt, gefolgt vom Leerzeichen und dem Cursor, auf dem Bildschirm erscheinen.)

Eine Datenbankdatei wird mit dem Befehl

```
CREATE <db-datei>
```

erstellt.

Der Dateiname darf bis zu 8 Zeichen lang sein und muss mit einem Buchstaben beginnen. Als Demonstrationsbeispiel erstellen wir eine Datenbankdatei namens "kunden".
Geben Sie bitte ein:

```
CREATE kunden <ET>
```

(Die <ET>-Taste muss zum Abschluss jedes Befehls gedrueckt werden. Im folgenden wird dies deshalb nicht mehr ausdruecklich erwaehnt.)

Soll die Datei auf einem anderen als dem Standardlaufwerk gespeichert werden, muessen Sie zusaetzlich zum Dateinamen noch die Bezeichnung des Laufwerkes mit anschliessendem Doppelpunkt angeben. Im obigen Fall also z.B. CREATE b:kunden. In all unseren Beispielen verzichten wir aber auf die Laufwerksangabe. (Sie erfuhren im Abschnitt 2.3.2., wie man mit SET DEFAULT das Laufwerk B dauerhaft zum Standardlaufwerk fuer die REDABAS-4 Dateneingabe und -ausgabe erklaeren kann.)

Durch den Befehl "CREATE kunden" erzeugt REDABAS-4 eine Datenbankdatei mit dem Namen "kunden.dbd". (Der Teil des Dateinamens nach dem Punkt ist der DCP-Dateityp, er wird von REDABAS-4 automatisch vergeben.)

REDABAS-4 ruft eine Routine zur Strukturierung der Datensaetze auf. Der Bildschirm wird geloescht, und auf dem Bildschirm erscheint die im Bild 4 gezeigte Eingabemaske.

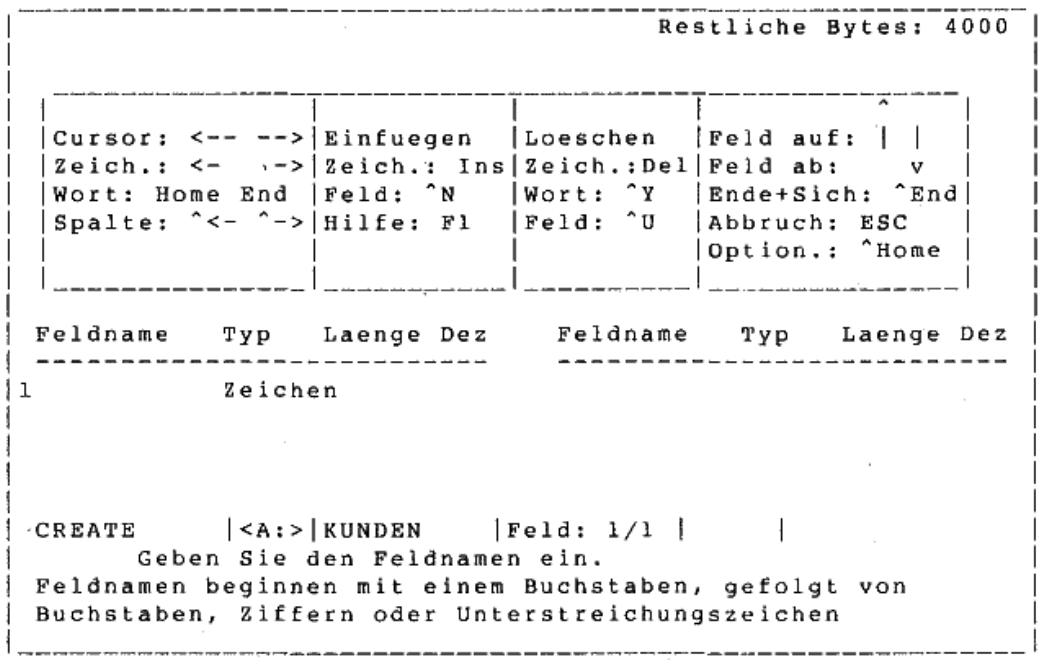



Bild 4 Eingabemaske fuer die Definition der Struktur einer Datenbankdatei

Rechts oben sagt Ihnen REDABAS-4, wieviele Bytes Sie fuer einen Datensatz (noch) zur Verfuegung haben. Im Abschnitt 1.1. erfuhren Sie, dass REDABAS-4 eine Datensatzlaenge von 4000 Bytes zulaesst. In unserem Fall moechten wir erst mit der Definition beginnen, entsprechend ist der angezeigte Wert.

In dem eingerahmten Feld im oberen Bildteil sind die Tastenbelegungen zur Cursorbewegung und fuer weitere moegliche Funktionen innerhalb dieser Eingabemaske erklaert. Anfangs werden Sie die Erklaerungen brauchen, spaeter oder auch falls Sie den Bildschirm vollstaendig fuer Feldeintragungen benoetigen, koennen Sie die Tastenerklaerungen ausblenden.

Wenn Sie die Funktionstaste <F1> druecken, verschwinden diese Hinweise, beim erneuten Druecken von <F1> erscheinen sie wieder auf dem Bildschirm. Zur Wahrung der Uebersichtlichkeit innerhalb dieser Schrift haben wir die Hilfstafeln in weiteren Beispielen nicht mehr mit dargestellt.

Der Hauptteil des Bildschirms steht nun fuer die eigentliche Definition zur Verfuegung (linke und rechte Bildschirmseite). Links ist schon Feldnummer 1 eingetragen. (REDABAS-4 vergibt diese Nummern automatisch.) Ein helles Kaestchen weist darauf hin, dass hier die erste Eintragung -naemlich der Name des ersten Datenfeldes- erfolgen soll.
Generell wird jedes Feld eines Datensatzes durch den Feldnamen, den Datentyp, die Feldlaenge und bei numerischen Feldern zusaetzlich durch die Anzahl der Dezimalstellen definiert.

Nachfolgend werden diese Angaben detaillierter beschrieben.

Feldname

Bezeichnung des Datenfeldes
max. 10 Zeichen, beginnend mit einem Buchstaben, danach sind auch Ziffern und Unterstreichungszeichen erlaubt, jedoch keine Leerzeichen, kein Doppelpunkt (im Unterschied zu REDABAS!) und keine Sonderzeichen.

Typ

Datentyp eines Datenfeldes
REDABAS-4 gestattet 5 unterschiedliche Datentypen:

C: Zeichen
N: Numerisch
L: Logisch
D: Datum
M: MERK

Der Datentyp wird entweder durch Betaetigen der Leertaste (bis der gewuenschte Typ erscheint) und Bestaetigen mit <ET> vereinbart oder durch Eingabe des Anfangsbuchstabens (Beachte Z fuer Zeichen!) festgelegt.

Zeichen

(alphanumerisch)
Alle druckbaren ASCII-Zeichen, einschliesslich der Ziffern, Sonderzeichen und des Leerzeichens.
Im allgemeinen enthalten Felder vom Datentyp "Zeichen" Text. Dieser Typ sollte aber auch fuer Datenfelder mit Zahlen verwendet werden, mit denen keine arithmetischen Operationen ausgefuehrt werden (z. B. PLZ und Tel.Nr.).
Der Typ "Zeichen" wird am haeufigsten verwendet, deshalb bietet REDABAS-4 "Zeichen" als Standard an. Die Eintragung "Zeichen" in der Eingabe- bzw. Editiermaske kann entweder mit <ET> bestaetigt, oder auch durch einen anderen der im folgenden angefuehrten Datentypen ueberschrieben werden.
Die maximale Laenge eines Zeichenfeldes ist 254 Zeichen.

Numerisch

Es sind nur Zahlen, der Dezimalpunkt und die Vorzeichen "+" und "-", jedoch kein Komma erlaubt.
Die maximale Laenge eines numerischen Feldes ist 19 Bytes; max. 15 Stellen sind nach dem Dezimalpunkt moeglich.

Logisch

Logische Datenfelder sind generell nur 1 Zeichen lang. Der Inhalt eines logischen Feldes kann aus den Abkuerzungen fuer True (Wahr) oder False (Falsch) bzw. Ja (Yes) oder Nein (No) bestehen.

Datum

Datumsfelder sind 8 Zeichen lang. Mit Werten vom Typ "Datum" kann man Berechnungen ausfuehren.

REDABAS-4 bietet fuer Datumsfelder zahlreiche Funktionen an.

MERK — Merkfelder nehmen in einer Datenbankdatei 10 Zeichen ein.
Intern koennen jedoch pro Merkfeld bis zu 4096 Zeichen gespeichert werden, d.h. es wird eine gesonderte Datei (Typ .DBM) zur Aufnahme dieser Merkfeldtexte angelegt. Dieses Merkfeld (wie auch Datum ein gegenueber REDABAS neuer Datenfeldtyp) ermoeglicht die Aufnahme eines individuellen Textabschnittes wie z.B. Bemerkungen, verbale Informationen. (Vgl. Abschn. 3.6.)

Laenge — Feldlaenge
Die Laenge eines Datenfeldes richtet sich nach dem Datentyp:
Zeichen: max. 254
Numerisch: max. 19
Bei Dezimalzahlen muss eine Stelle fuer den Dezimalpunkt beruecksichtigt werden und eine weitere Stelle, wenn Vorzeichen gewuenscht werden.
Logisch: REDABAS-4 vergibt automatisch eine Laenge von 1 Zeichen.
Datum: REDABAS-4 vergibt automatisch eine Laenge von 8 Zeichen.
MERK: REDABAS-4 vergibt automatisch eine Laenge von 10 Zeichen.

Dez — Angabe, die REDABAS-4 nur bei numerischem Datentyp erwartet: Stellenzahl nach dem Dezimalpunkt, wenn Dezimalstellen gewuenscht werden.

Wir moechten Ihnen an dieser Stelle noch einige allgemeine Hinweise fuer die Festlegung der Struktur einer Datenbankdatei geben:

- Beachten Sie bei der Definition der Feldlaenge, dass die Felder einerseits so gross sein muessen, dass die laengste Eintragung darin Platz findet, andererseits aber fuer eine rationelle Speicherung und Zugriffszeit die Feldlaengen moeglichst klein sein sollen. Waehlen Sie gegebenenfalls mit eindeutigen Abkuerzungen ein Optimum.
  Bei numerischen Feldern muss bei der Festlegung der Feldlaenge beachtet werden, dass kein Felduberlauf durch arithmetische Operationen auftreten kann.

- Datenbankdateien, die Sie gemeinsam verarbeiten moechten, muessen ein gemeinsames Datenfeld besitzen (wie z.B. Kundennummer).

- Merkfelder koennen nicht durchsucht, sortiert, indiziert oder einer Variablen zugewiesen werden. Waehlen Sie also fuer auszuwertende Informationen nicht den Datentyp "MERK", sondern "Zeichen".

- Bemerken Sie bereits nach Abschluss der Eingabe Ihrer Dateistruktur in die Eingabemaske (vgl. Bild 3) oder aber waehrend der spaeteren Arbeit mit der Datei, dass die Struktur nicht (mehr) Ihren Erfordernissen entspricht, kann diese mit dem in Abschnitt 3.3. beschriebenen Befehl MODIFY STRUCTURE jederzeit geaendert werden.

Im unteren Teil des Bildschirms (vgl. Bild 4) sind Status-, Anweisungs- und Meldungszeile ausgewiesen, die im Abschn. 2.5. erklaert sind. In der Statuszeile werden nach "Feld:" die in der Eingabemaske aktuelle Feldnummer und die Anzahl der aktuell definierten Felder angezeigt.

Definieren Sie jetzt die Datei "kunden" wie in Bild 5 dargestellt.

```
                                          Restliche Bytes: 3955

  Feldname  Typ      Laenge Dez   Feldname   Typ  Laenge Dez
  ---------------------------     -------------------------
1 NAME      Zeichen   10
2 VORNAME   Zeichen    8
3 PLZ       Zeichen    4
4 ORT       Zeichen    8
5 STRASSE   Zeichen   15
```

Bild 5 Struktur der Datenbankdatei "kunden"

Sie geben die Informationen zu jedem Feld Zeile fuer Zeile ein. Haben Sie den ersten Feldnamen "name" eingegeben, druecken Sie <ET>. Der Cursor rueckt automatisch auf die Standardeintragung fuer Typ "Zeichen", die Sie mit <ET> bestaetigen.

Das Feld "name" soll 10 Zeichen lang sein, also geben Sie bei Laenge die Zahl 10 ein, die wieder mit <ET> abgeschlossen wird. Sie brauchen die Zahl nicht rechtsbuendig einzutragen, durch Druecken von <ET> geschieht dies automatisch. Bei einem Feld vom Typ "Zeichen" erwartet REDABAS-4 keine weitere Eintragung, deshalb springt der Cursor sofort auf die naechste Zeile, wo das zweite Datenfeld definiert wird. Sie fahren in dieser Weise fort, bis alle 5 Datenfelder definiert sind. Steht der Cursor in Zeile 6, druecken Sie <CTRL-End> oder die <ET>-Taste. REDABAS-4 erkennt dann, dass Sie kein weiteres Feld anfuegen moechten. Wenn Sie erneut <ET> druecken, wird die Definition der Struktur abgeschlossen und die Struktur der noch leeren Datei "kunden" wird gespeichert. Wir haben alle 5 Datenfelder alphanumerisch, d. h. mit Datentyp "Zeichen" definiert. Beachten Sie, dass auch das Feld PLZ (Postleitzahl) als Typ "Zeichen" definiert wurde, obwohl es normalerweise aus Ziffern besteht. Da keinerlei arithmetische Operationen damit ausgefuehrt werden, ist dies moeglich.

Vergleichen Sie Bild 5. Von 128 moeglichen Feldern haben wir 5 spezifiziert, deren Gesamtlaenge 45 (von max. 4000) betraegt. In der rechten oberen Bildschirmecke sehen Sie eine dementsprechende Ausschrift.

Wenn Sie waehrend des Eintragens in die Eingabemaske Korrekturen vornehmen wollen, ist dies ohne weiteres moeglich.

Die Hilfstafel (Sie koennen sie jederzeit durch Druecken der <F1>-Taste ein- bzw. ausblenden) gibt Ihnen ueber die Cursorbewegung und weitere Moeglichkeiten Auskunft. Solange Sie das die Definition abschliessende <ET> nicht gedrueckt haben, koennen Sie beliebig veraendern.

## Erfassen von Daten

Sofort nach Abschluss der Definition der Struktur fragt Sie REDABAS-4

Wollen Sie jetzt Datensaetze eingeben? (J/N)

Wenn Sie jetzt keine Aenderung mit MODIFY STRUCTURE vornehmen wollen, antworten Sie mit "J" (Ja).
REDABAS-4 loescht den Bildschirm, und Sie beginnen mit der Eingabe Ihrer Daten. REDABAS-4 zeigt Ihnen eine Maske zur Datenerfassung entsprechend der definierten Struktur:

```
 NAME    :           :
 VORNAME :         :
 PLZ     :     :
 ORT     :         :
 STRASSE :               :
```

Bild 6 Maske fuer die Datenerfassung

Auf jeder Zeile der Eingabemaske steht der Feldname, und dahinter hell unterlegt befindet sich ein leeres Feld in der definierten Laenge. Dort tragen Sie die Daten ein. Sobald Sie das Feld vollstaendig ausgefuellt oder <ET> gedrueckt haben, rueckt der Cursor auf das naechste Feld. Fuer Korrekturen koennen Sie den Cursor wieder (wie in der Hilfstafel angezeigt) beliebig bewegen, Sie koennen auch "Einfuegen" mit der <Ins>-Taste bzw. der Tastenkombination <CTRL-V> waehlen oder "Loeschen" mit der <Del>-Taste.

Bitte achten Sie bei der Dateneingabe auf Gross- und Kleinschreibung, da bei spaeteren Auswertungen der Datenbank auch dies fuer Unterscheidungen herangezogen wird.

Wenn Ihnen bei der Eingabe numerischer oder logischer Feldinhalte sowie bei Datumsangabe formale Fehler unterlaufen, weist REDABAS-4 die falsche Eingabe ab und macht Sie durch einen Warnton darauf aufmerksam.

Nachdem das letzte Feld (im Beispiel "strasse") in einem Datensatz ausgefuellt wurde, wird die <ET>-Taste betaetigt, und es kann mit der Eingabe des naechsten Datensatzes begonnen werden.

Von REDABAS-4 wird automatisch eine fortlaufende Satznummer vergeben, hier beginnend mit 1 (Anfang der Datei). Sie ist in der Statuszeile sichtbar. Pro Datei koennen theoretisch bis zu einer Milliarde Datensaetze gespeichert werden (d.h. wenn der externe Speicherplatz ausreicht).

Geben Sie bitte die folgenden Datensaetze der Datenbankdatei "kunden" ein:

```
Lehmann    Kurt    8060 DRESDEN    Landstr. 15
Schmidt    Gisela  8223 THARANDT   Bahnhofstr. 43
Moser      Otto    8212 FREITAL    Neugraben 8
Weber      Franz   8223 THARANDT   Hortgasse 14
Schulze    Ernst   8060 DRESDEN    Nebenstr. 61
Miller     Peter   8212 FREITAL    Nelkenweg 23
Meier      August  8122 RADEBEUL   Uferstr. 12
Neumann    Inge    8222 RABENAU    Gartenstr. 4
Fleischer  Karl    8212 FREITAL    Am See 83
Schmidt    Heinz   8223 THARANDT   Waldweg 42
```

Sie haben jetzt 10 Datensaetze eingegeben. Sie beenden die Erfassung durch Druecken der <ET>-Taste, wenn die Maske fuer den 11. Satz angezeigt wird. (<CTRL-End> wuerde einen leeren 11. Satz liefern.)

Mit der Eingabe dieser Saetze hat REDABAS-4 sie bereits auf der Diskette gespeichert. Wollen Sie jetzt Ihre Arbeit beenden, geben Sie den Befehl QUIT ein.

## Protokollieren

Moechten Sie weiterarbeiten, ist es zweckmaessig, eine Liste Ihrer bisherigen Eingaben anzufertigen. Sie benoetigen dazu die Befehle

```
USE <db-datei>
LIST
```

Mit dem Befehl USE wird eine bereits vorhandene Datenbankdatei aktiviert. Immer dann, wenn Sie irgendeine Bearbeitung an einer Datei vornehmen moechten, muss die Datei -sofern sie nicht schon eroeffnet ist- mit USE eroeffnet werden. Die Datei bleibt so lange verfuegbar, bis ein neuer USE-Befehl eine andere Datenbankdatei aktiviert. (In den folgenden Beispielen setzen wir voraus, dass die Datei bereits eroeffnet ist.)
Sie geben bitte ein:

```
USE kunden
LIST
```

Wir erhalten eine Liste aller Datensaetze, versehen mit einer von REDABAS-4 vergebenen laufenden Nummer.

```
:USE kunden
:LIST
Satznr.  NAME       VORNAME PLZ  ORT         STRASSE
      1  Lehmann    Kurt    8060 DRESDEN     Landstr. 15
      2  Schmidt    Gisela  8223 THARANDT    Bahnhofstr. 43
      3  Moser      Otto    8212 FREITAL     Neugraben 8
      4  Weber      Franz   8223 THARANDT    Hortgasse 14
      5  Schulze    Ernst   8060 DRESDEN     Nebenstr. 61
      6  Miller     Peter   8212 FREITAL     Nelkenweg 23
      7  Meier      August  8122 RADEBEUL    Uferstr. 12
      8  Neumann    Inge    8222 RABENAU     Gartenstr. 4
      9  Fleischer  Karl    8212 FREITAL     Am See 83
     10  Schmidt    Heinz   8223 THARANDT    Waldweg 42
```

Bild 7 Liste der Datenbankdatei "kunden"

Wollen Sie die Liste auf dem Drucker ausgeben, betaetigen Sie die <CTRL-P>-Tastenkombination und wiederholen Sie den Befehl LIST. Das Abschalten des Druckers erfolgt durch nochmaliges Druecken von <CTRL-P>.

Sie koennen den LIST-Befehl aber auch um den Zusatz TO PRINT erweitern, wenn Sie sich fuer eine Ausgabe auf dem Drucker entscheiden.

Mit LIST koennen Sie sich auch die Struktur Ihrer Datenbankdatei anzeigen lassen, Sie erweitern den Befehl auf

```
LIST STRUCTURE
```

```
: LIST STRUCTURE

Datenbankdateistruktur : A:kunden.dbd
Anzahl der Datensaetze :     10
Letztes Aenderungsdatum: 13.08.89
Feld  Feldname     Typ            Laenge       Dez
   1  NAME         Zeichen            10
   2  VORNAME      Zeichen             8
   3  PLZ          Zeichen             4
   4  ORT          Zeichen             8
   5  STRASSE      Zeichen            15
* Gesamt *                           46
```

Bild 8 Struktur der Datenbankdatei "kunden"

Es bedeuten:

Datenbankdateistruktur - A:kunden.dbd
Der Dateityp "dbd" weist darauf hin, dass es sich um eine Datenbankdatei handelt.

Letztes Aenderungsdatum - 13.08.89
Das Datum einer Aenderung kann das Aktualisieren sowohl der Daten als auch der Dateistruktur betreffen.

* GESAMT * bezieht sich auf die Gesamtlaenge eines Datensatzes. Es wurden die einzelnen Feldlaengen addiert und ein Zeichen hinzugefuegt, dessen Bedeutung spaeter unter der Ueberschrift "Loeschen von Saetzen" erklaert wird.

Es ist unbedingt zu empfehlen, die Dateistruktur auszudrucken, da Sie zu jeder Zeit ueber Namen und Typ der Datenfelder wegen spaeter auszufuehrender Operationen Bescheid wissen muessen.

### 3.1.2. Eingeben weiterer Datensaetze

Fuer das An- bzw. Einfuegen weiterer Datensaetze stehen die Befehle APPEND und INSERT zur Verfuegung.

APPEND

ermoeglicht das Anfuegen von Saetzen an eine vorhandene Daten-

bankdatei. Haben Sie APPEND eingegeben, antwortet REDABAS-4 mit dem Anzeigen der Eingabemaske.

Im Beispiel moechten wir die Datei "kunden" um drei Kunden erweitern.
Wir geben ein:

```
APPEND
```

Es erscheint die Eingabemaske (bei voreingestellten SET STATUS OFF):

```
Satz-Nr.      11
NAME     :          :
VORNAME  :         :
PLZ      :     :
ORT      :         :
STRASSE  :                :
```

Bild 9 Maske fuer das Anfuegen eines neuen Datensatzes

Geben Sie die folgenden 3 Datensaetze ein:

```
Lolle    Peter     8122 RADEBEUL      Goethestr. 13
Heinze   Max       8060 DRESDEN       Rosenstr. 78
Tauber   Fred      8028 DRESDEN       Fliederstr. 42
```

Bestaetigen Sie den zuletzt eingegebenen Wert fuer "strasse" mit <CTRL-End>. Falls Sie das verpasst und <ET> gedrueckt haben, druecken Sie <Esc> (oder nochmals <ET>).

Die Datenbankdatei "kunden" enthaelt damit 13 Saetze. Wir listen nun die vollstaendige Datei auf:

```
: LIST
 Satznr.  NAME       VORNAME PLZ   ORT        STRASSE
       1  Lehmann    Kurt    8060  DRESDEN    Landstr. 15
       2  Schmidt    Gisela  8223  THARANDT   Bahnhofstr. 43
       3  Moser      Otto    8212  FREITAL    Neugraben 8
       4  Weber      Franz   8223  THARANDT   Hortgasse 14
       5  Schulze    Ernst   8060  DRESDEN    Nebenstr. 61
       6  Miller     Peter   8212  FREITAL    Nelkenweg 23
       7  Meier      August  8122  RADEBEUL   Uferstr. 12
       8  Neumann    Inge    8222  RABENAU    Gartenstr. 4
       9  Fleischer  Karl    8212  FREITAL    Am See 83
      10  Schmidt    Heinz   8223  THARANDT   Waldweg 42
      11  Lolle      Peter   8122  RADEBEUL   Goethestr. 13
      12  Heinze     Max     8060  DRESDEN    Rosenstr. 78
      13  Tauber     Fred    8028  DRESDEN    Fliederstr. 42
```

Bild 10 Erweiterte Datenbankdatei "kunden"

Der Befehl

```
INSERT
```

ermoeglicht das Einfuegen von Saetzen an jeder beliebigen Stelle der Datenbankdatei. Fuer das vorausgehende Positionieren auf einen bestimmten Datensatz gibt es verschiedene Moeglich-

keiten Das Einfachste ist es, anstelle eines Befehls die Datensatznummer einzugeben (z.B.: 4). Der einzufuegende Satz kann vor oder nach dem aktuellen Satz eingeordnet werden, vgl. Abschnitt 9.3.3.).

### 3.1.3. Aendern von Datensaetzen

Ergeben sich Veraenderungen an bereits erfassten und abgespeicherten Datensaetzen, bietet REDABAS-4 drei Moeglichkeiten der Aenderung. Die Befehle lauten:

```
EDIT
CHANGE
REPLACE
```

Zunaechst moechten wir Ihnen anhand des Demonstrationsbeispiels die Wirkungsweise von EDIT vorfuehren.

Es ist die Anschrift des Kunden Franz Weber, Datensatz Nr. 4, zu veraendern.
Geben Sie ein:

```
EDIT 4
```

Darauf erscheint die schon von der Datenerfassung bekannte Maske mit den Daten von Datensatz Nr. 4.

Mittels der Cursortasten koennen Sie alle Zeichenpositionen und Datenfelder erreichen und diese wunschgemaess ueberschreiben.

```
Satz-Nr.      4
NAME       :Weber     :
VORNAME    :Franz   :
PLZ        :8223:
ORT        :THARANDT:
STRASSE    :Hortgasse 14    :
```

Bild 11 (Unveraenderter) Datensatz Nr. 4

Positionieren Sie den Cursor auf das Datenfeld "strasse" und aendern Sie dieses wie folgt:

```
Satz-Nr.      4
NAME       :Weber     :
VORNAME    :Franz   :
PLZ        :8223:
ORT        :THARANDT:
STRASSE    :Heideweg 66     :
```

Bild 12 Veraenderter Datensatz Nr. 4

Mit <CTRL-W> bzw. <CTRL-End> wird die Bearbeitung des Datensatzes abgeschlossen. Der veraenderte Datensatz wird damit gespeichert.

Wenn man nach dem 4. Datensatz eine Reihe weiterer Datensaetze aendern moechte, ist das ohne erneutes EDIT moeglich. Anstelle der o. a. abschliessenden Tastenkombinationen wird die <ET>-Taste gedrueckt, und die Maske von Satz 5 erscheint. So koennen Sie Satz fuer Satz aendern.

Sie koennen aber auch die Tasten <PgDn> (Page Down - Seite vor) oder <PgUp> (Page Up - Seite zurueck) fuer das Positionieren

auf andere Datensaetze nutzen.

Wir moechten zunaechst keine weiteren Aenderungen vornehmen und uns die Datei "kunden" nach der Aenderung nochmals mit LIST ansehen:

```
: LIST
Satznr. NAME       VORNAME  PLZ   ORT       STRASSE
      1 Lehmann    Kurt     8060  DRESDEN   Landstr. 15
      2 Schmidt    Gisela   8223  THARANDT  Bahnhofstr. 43
      3 Moser      Otto     8212  FREITAL   Neugraben 8
      4 Weber      Franz    8223  THARANDT  Heideweg 66
      5 Schulze    Ernst    8060  DRESDEN   Nebenstr. 61
      6 Miller     Peter    8212  FREITAL   Nelkenweg 23
      7 Meier      August   8122  RADEBEUL  Uferstr. 12
      8 Neumann    Inge     8222  RABENAU   Gartenstr. 4
      9 Fleischer  Karl     8212  FREITAL   Am See 83
     10 Schmidt    Heinz    8223  THARANDT  Waldweg 42
     11 Lolle      Peter    8122  RADEBEUL  Goethestr. 13
     12 Heinze     Max      8060  DRESDEN   Rosenstr. 78
     13 Tauber     Fred     8028  DRESDEN   Fliederstr. 42
```

Bild 13 Datenbankdatei "kunden" nach der Aenderung

Im folgenden geben wir Ihnen einen Ueberblick ueber die innerhalb der Editiermasken wichtigen Tasten bzw. Tastenkombinationen und deren Funktionen. (Die Kombination besteht immer aus der <CTRL>-Taste und einer Buchstabentaste, wobei die Buchstabentaste bei niedergedrueckter <CTRL>-Taste zu betaetigen ist.)
Mit diesen Tasten bzw.Tastenkombinationen koennen Sie Ihre Daten aehnlich wie in einem Textverarbeitungssystem bearbeiten.

| | |
|---|---|
| <CTRL-W><br><CTRL-End> | Abschliessen der Funktion EDIT |
| <CTRL-C><br><PgDn> | Speichern des (geaenderten) Datensatzes und Aufrufen des naechsten Satzes zur Bearbeitung |
| <CTRL-R><br><PgUp> | Speichern des (geaenderten) Datensatzes und Aufrufen des vorhergehenden Satzes zur Bearbeitung |
| <Esc><br><CTRL-Q> | Abbruch der Aenderungen im gerade bearbeiteten Datensatz und Verlassen der EDIT-Funktion |
| <DEL><br><CTRL-G> | Loeschen des durch den * Cursor positionierten Zeichens |
| <CTRL-T> | Loeschen des Wortes ab Cursorposition |
| <CTRL-Y> | Loeschen des Feldes ab Cursorposition |
| Ruecktaste | Loeschen des Zeichens links vom Cursor |

| | |
|---|---|
| <CTRL-U> | Ein-/Ausschalten der Loeschmarkierung fuer einen Datensatz (vergl. Abschnitt 3.1.4.). Beim Einschalten erscheint am oberen Rand der Satzmarke oder in der Statuszeile "Del". Drueckt man <CTRL-U> nochmals, kann die Loeschung rueckgaengig gemacht werden. |
| <Ins><br><CTRL-V> | Umschalten zwischen Ueberschreiben und Einfuegen von Zeichen<br>Wird Einfuegen gewaehlt, wird das auf dem Bildschirm rechts oben oder in der Statuszeile angezeigt. |
| Pfeiltaste oben<br><CTRL-E> | Bewegen des Cursors eine Zeile nach oben |
| Pfeiltaste unten<br><CTRL-X> | Bewegen des Cursors eine Zeile nach unten |
| Pfeiltaste <-<br><CTRL-S> | Bewegen des Cursors ein Zeichen nach links |
| Pfeiltaste ->  <br><CTRL-D> | Bewegen des Cursors ein Zeichen nach rechts |
| <Home><br><CTRL-A> | Bewegen des Cursors zum Wortanfang bzw. ein Wort nach links (oben) |
| <End><br><CTRL-F> | Bewegen des Cursors zum Wortende bzw. ein Wort nach rechts (unten) |
| <ET> | Bewegen des Cursors zum naechsten Feld |

Eine zweite Moeglichkeit der Veraenderung von Datensaetzen ist die Anwendung des Befehls

CHANGE <bereich> FIELDS <feldfolge>

Mit geringem Aufwand koennen Sie den Inhalt einzelner Datenfelder aendern. Nach der Eingabe des entsprechend formulierten Befehls wird die Satznummer, der Name (die Namen) des (der) zu aendernden Feldes (Felder) und der zugehoerige Feldinhalt angezeigt, der beliebig ueberschrieben werden kann.(Vergleichen Sie bitte fuer Symboldefinitionen wie <bereich>, <feldfolge> Abschnitt 9.1. dieser Programmtechnischen Beschreibung.)

Die Aenderung eines Datenfeldes demonstrieren wir Ihnen wieder an der Kundendatei im Bild 14.

Der Kunde "Schulze" ist von der Nebenstrasse auf die Hauptstrasse verzogen. Es ist also das Feld "strasse" zu veraendern.

```
: LIST FOR name="Schulze"
 Satznr. NAME      VORNAME    PLZ    ORT        STRASSE
       5 Schulze   Ernst      8060   DRESDEN    Nebenstr. 61
: CHANGE RECORD 5 FIELDS strasse

Satz Nr.      5

STRASSE     Nebenstr. 61      (vor der Aenderung)

STRASSE     Hauptstr. 61      (nach der Aenderung)
```

Bild 14  Aendern eines Datensatzes mit CHANGE

CHANGE ohne Eintragung ermoeglicht das Aendern aller Felder und Datensaetze.
Weiterhin koennen Sie einige oder auch alle Datensaetze mit dem Befehl

```
REPLACE <feld> WITH <ausdr>
```

aendern.

Dieser Befehl ist sehr maechtig und kann folgendermassen interpretiert werden:
Ersetze den Inhalt des Feldes <feld> durch den Inhalt oder das Ergebnis von <ausdr>.
<ausdr> kann ein anderer Feldname, ein Text, eine Zahl oder eine Formel sein. (Vergleichen Sie hierzu Abschnitt 9.1. "Symboldefinitionen".)

Will man das "Ersetzen" an allen Saetzen einer Datenbankdatei vornehmen, wird der Befehl erweitert:

```
REPLACE ALL <feld> WITH <ausdr>
```

Diese grosszuegige Moeglichkeit der Manipulierung einer ganzen Datei bietet nur ein relationales Datenbankbetriebssystem. Sie werden bei der weiteren Arbeit mit dieser Programmtechnischen Beschreibung noch die Anwendung des Befehls REPLACE kennenlernen.

### 3.1.4. Loeschen von Datensaetzen

Sollen bestimmte Datensaetze aus der Datenbankdatei geloescht werden, kann dies in zwei Varianten erfolgen.

Die erste Moeglichkeit besteht in der Anzeige des zu loeschenden Satzes mit dem EDIT-Befehl und dem Eintragen der Markierung zum Loeschen durch <CTRL-U>. Am oberen Bildschirmrand oder in der Statuszeile erscheint der Hinweis "Del".

Die zweite Moeglichkeit der Loeschung von Datensaetzen bietet der Befehl

```
DELETE
```

Mit diesem Befehl koennen Sie einen Datensatz unter Angabe der Satznummer mit dem Loeschkennzeichen versehen. Ausserdem ist es moeglich, einen Datensatzbereich oder auch alle Datensaetze anzugeben oder Auswahlbedingungen zu spezifizieren.

Nach beiden Loeschvarianten bleiben die Datensaetze physisch erhalten. Sie werden an einer dafuer reservierten Stelle im Datensatz (von der schon vorn gesprochen wurde) mit der Markierung "*" versehen.

Wir moechten Ihnen anhand unseres Beispiels das Loeschen von Datensaetzen demonstrieren.
Im Bild 15 sehen Sie die Datei "kunden" aufgelistet, wobei die Datensaetze Nr. 6, 8 und 11 durch folgende Befehle markiert worden sind:

```
USE kunden
DELETE RECORD 6
DELETE RECORD 8
DELETE RECORD 11
```

```
: LIST
Satznr.    NAME       VORNAME  PLZ   ORT        STRASSE
      1    Lehmann    Kurt     8060  DRESDEN    Landstr. 15
      2    Schmidt    Gisela   8223  THARANDT   Bahnhofstr. 43
      3    Moser      Otto     8212  FREITAL    Neugraben 8
      4    Weber      Franz    8223  THARANDT   Heideweg 66
      5    Schulze    Ernst    8060  DRESDEN    Hauptstr. 61
      6 *  Miller     Peter    8212  FREITAL    Nelkenweg 23
      7    Meier      August   8122  RADEBEUL   Uferstr. 12
      8 *  Neumann    Inge     8222  RABENAU    Gartenstr. 4
      9    Fleischer  Karl     8212  FREITAL    Am See 83
     10    Schmidt    Heinz    8223  THARANDT   Waldweg 42
     11 *  Lolle      Peter    8122  RADEBEUL   Goethestr. 13
     12    Heinze     Max      8060  DRESDEN    Rosenstr. 78
     13    Tauber     Fred     8028  DRESDEN    Fliederstr. 42
```

Bild 15 Datenbankdatei "kunden" mit den zur Loeschung markierten Datensaetzen

Alle Saetze sind jetzt noch vorhanden, die Loeschung koennte jederzeit mit dem Befehl

```
RECALL ALL
```

rueckgaengig gemacht werden. Die physische Loeschung der Saetze erfolgt erst nach Eingeben des Befehls

```
PACK
```

der auch eine neue fortlaufende Vergabe der Satznummern bewirkt.
(Fuehren Sie aber bitte diesen Befehl jetzt nicht aus!)

Bild 16 zeigt Ihnen die Datei "kunden", nachdem die Loeschung erfolgt ist. Die verbliebenen Datensaetze sind mit neuen fortlaufenden Nummern abgespeichert.

```
: PACK

10 Saetze kopiert

: LIST
Satznr.   NAME       VORNAME   PLZ    ORT       STRASSE
      1   Lehmann    Kurt      8060   DRESDEN   Landstr. 15
      2   Schmidt    Gisela    8223   THARANDT  Bahnhofstr.43
      3   Moser      Otto      8212   FREITAL   Neugraben 8
      4   Weber      Franz     8223   THARANDT  Heideweg 66
      5   Schulze    Ernst     8060   DRESDEN   Hauptstr. 61
      6   Meier      August    8122   RADEBEUL  Uferstr. 12
      7   Fleischer  Karl      8212   FREITAL   Am See 83
      8   Schmidt    Heinz     8223   THARANDT  Waldweg 42
      9   Heinze     Max       8060   DRESDEN   Rosenstr. 78
     10   Tauber     Fred      8028   DRESDEN   Fliederstr. 42
```

Bild 16 Drei markierte Saetze der Datenbankdatei "kunden" sind physisch geloescht

Sollen alle Datensaetze einer aktivierten Datenbankdatei geloescht werden, geschieht das mit dem Befehl

```
ZAP
```

Dabei bleibt die Struktur der Datei erhalten und die Datei selbst geoeffnet.

## 3.2. Wiederauffinden aus der Datenbank

Bei der praktischen Nutzung einer Datenbank kommt es vor allem darauf an, nach der Phase der Einrichtung der Datenbank gezielte Zugriffe fuer die Befriedigung unterschiedlicher Informationsbeduerfnisse vorzunehmen.

Sie lernten bereits den Befehl LIST fuer das Auflisten Ihrer Datenbankdatei kennen. Dieser Befehl kann, um einige Angaben erweitert, die Datei jedoch auch nach bestimmten Kriterien durchsuchen und/oder nur ausgewaehlte Teile der Datei, spezifizierte Datensaetze oder Felder anzeigen.

### 3.2.1. Selektion bestimmter Datenfelder

Zunaechst moechten wir von der Datenbankdatei "kunden" nur die Felder "name", "vorname" und "ort" ausgeben lassen.
Bitte geben Sie ein

```
LIST Vorname,Name,Ort
```

Es werden nun nur die spezifizierten Felder ausgegeben, die nicht angefuehrten Felder werden unterdrueckt. Es ist wichtig, dass Sie die Feldnamen genau so schreiben, wie sie definiert wurden (auf Gross- und Kleinbuchstaben brauchen Sie bei den Feldnamen nicht zu achten; allerdings werden im LIST-Befehl explizit spezifizierte Feldnamen als Spaltenueberschrift nicht in Grossbuchstaben ausgegeben, sondern so, wie in der Befehlszeile geschrieben!).

```
: LIST Vorname,Name,Ort

 Satznr.    Vorname        Name                Ort
       1    Kurt           Lehmann             DRESDEN
       2    Gisela         Schmidt             THARANDT
       3    Otto           Moser               FREITAL
       4    Franz          Weber               THARANDT
       5    Ernst          Schulze             DRESDEN
       6    Peter          Miller              FREITAL
       7    August         Meier               RADEBEUL
       8    Inge           Neumann             RABENAU
       9    Karl           Fleischer           FREITAL
      10    Heinz          Schmidt             THARANDT
      11    Peter          Lolle               RADEBEUL
      12    Max            Heinze              DRESDEN
      13    Fred           Tauber              DRESDEN
```

Bild 17 Datenbankdatei "kunden", spezifiziert nach Vorname, Name, Ort

### 3.2.2. Bedingte Auflistung von Datensaetzen

Will man die Auswertung einer Datei in Abhaengigkeit von der Erfuellung bestimmter Bedingungen vornehmen, geschieht dies mit

```
LIST FOR <bedingung>
```

Der einfachste Fall einer Bedingung ist, eine konstante Zeichenreihe vorzugeben und die Datei nach dieser Zeichenreihe zu durchsuchen.

#### Datenfeld als Suchbegriff

Im folgenden Beispiel soll die Kundendatei nach den Kunden durchsucht werden,
1. deren Wohnort "DRESDEN" ist:
2. deren Name "Schmidt" ist:

```
: LIST FOR ort = "DRESDEN"

Satznr.   NAME      VORNAME   PLZ     ORT        STRASSE
      1   Lehmann   Kurt      8060    DRESDEN    Landstr. 15
      5   Schulze   Ernst     8060    DRESDEN    Hauptstr. 61
     12   Heinze    Max       8060    DRESDEN    Rosenstr. 78
     13   Tauber    Fred      8028    DRESDEN    Fliederstr. 42


: LIST FOR name = "Schmidt"

Satznr.   NAME      VORNAME   PLZ     ORT        STRASSE
      2   Schmidt   Gisela    8223    THARANDT   Bahnhofstr. 43
     10   Schmidt   Heinz     8223    THARANDT   Waldweg 42
```

Bild 18 Datenbankdatei "kunden", in zwei Varianten durchsucht

Beachten Sie bei der Formulierung bitte folgende Regeln:

- Der Name des zu untersuchenden Datenfeldes darf gross oder klein geschrieben sein, muss aber bezueglich Zeichenfolge mit der Definition entsprechend der Dateistruktur uebereinstimmen.

- Die Zeichenreihe, die mit dem Inhalt der Datenfelder verglichen werden soll, muss neben genauer Uebereinstimmung in der Schreibweise auch bezueglich Gross- und Kleinschreibung mit den gespeicherten Daten uebereinstimmen. (Im Beispiel wurden in der Kundendatei die Ortsnamen mit Grossbuchstaben geschrieben, waere hier nach "Dresden" gesucht worden, waere kein Datensatz gefunden worden.)

- Direkt angegebene Zeichenreihen (alphanumerische Angaben, d.h. vom Datentyp "Zeichen") sind in Anfuehrungszeichen (einfach oder doppelt) zu setzen, rein numerische Werte werden ohne Anfuehrungszeichen geschrieben. Ob ein Datenfeld numerisch oder alphanumerisch ist, haben Sie bereits bei der Definition der Satzstruktur festgelegt, auch hier muessen die Angaben uebereinstimmen.

- Fuer numerische und alphanumerische Daten sowie Datumswerte gleichermassen gelten die aus der Mathematik bekannten Vergleichsoperatoren und deren Symbole:

```
        =    gleich                 <=   kleiner oder gleich
        <    kleiner                >=   groesser oder gleich
        >    groesser
# oder <>    ungleich
```

  Ein wichtiger Unterschied besteht bei der Abfrage zwischen Zahlen vom Typ "Zeichen" und vom Typ "Numerisch". Alphanumerische Zahlen sind eine Folge einzelner Ziffern ohne Beachtung des Stellenwertes. Die Ziffernfolge 82 ist also beispielsweise groesser als die Ziffernfolge 39876, da die erste Ziffer des ersten Wertes "8" groesser ist als die erste Ziffer des zweiten Wertes "3". Das erklaert auch die Tatsache, dass es bei dem folgenden Beispiel ausreicht, fuer plz = '82' anzugeben, um alle Kunden in den

Orten zu erhalten, deren Postleitzahl mit "82" beginnt.

Bild 19 demonstriert das Abfragen der Datei "kunden" mit dem Vergleichsoperator "gleich". Es soll nach Kunden in bestimmten Kreisen des Bezirkes Dresden (Postleitzahl 82xx) gesucht werden.

Bild 20 demonstriert die Anwendung des Vergleichsoperators "kleiner". Es werden alle Kunden aufgelistet, die in der Stadt Dresden wohnen. Da die Ziffernfolge "80xx" der Postleitzahl der Stadt Dresden entspricht, ist es moeglich, nach PLZ < 8100 zu fragen. Lautete die Bedingung LIST FOR plz = '80', erhielte man in diesem Beispiel dasselbe Ergebnis.

```
: LIST FOR plz = "82"

Satznr.   NAME       VORNAME   PLZ   ORT        STASSE
      2   Schmidt    Gisela    8223  THARANDT   Bahnhofstr. 42
      3   Moser      Otto      8212  FREITAL    Neugraben 8
      4   Weber      Franz     8223  THARANDT   Heideweg 66
      6   Miller     Peter     8212  FREITAL    Nelkenweg 23
      8   Neumann    Inge      8222  RABENAU    Gartenstr. 4
      9   Fleischer  Karl      8212  FREITAL    Am See 83
     10   Schmidt    Heinz     8223  THARANDT   Waldweg 42
```

Bild 19 Datenbankdatei "kunden", mit Vergleichsoperator "=" durchsucht

```
: LIST FOR plz < "8100"

Satznr.   NAME     VORNAME   PLZ   ORT      STRASSE
      1   Lehmann  Kurt      8060  DRESDEN  Landstr. 15
      5   Schulze  Ernst     8060  DRESDEN  Hauptstr. 61
     12   Heinze   Max       8060  DRESDEN  Rosenstr. 78
     13   Tauber   Fred      8028  DRESDEN  Fliederstr. 42
```

Bild 20 Datenbankdatei "kunden", mit Vergleichsoperator "<" durchsucht

## Feldausschnitt als Suchbegriff

Mit dem Teilzeichenreihen-Operator kann die Datei nach beliebigen Ausschnitten innerhalb eines Datenfeldes vom Typ "Zeichen" durchsucht werden. Beispielsweise kann die Kundendatei nach allen Personen durchsucht werden, in deren Namen ein "m" vorkommt. Der LIST-Befehl wird folgendermassen formuliert, wenn man mit dem Teilzeichenreihen-Operator arbeiten moechte:

```
LIST FOR <ausdrC>$<feld>
```

Der Teilzeichenreihen-[Vergleichs-]Operator kann nur auf Felder vom Typ "Zeichen" angewandt werden. <ausdrC> - ein Ausdruck vom Typ "Zeichen" - ist im einfachsten Falle eine konstante, d.h. direkt vorgegebene Zeichenreihe, die wieder durch Anfuehrungszeichen begrenzt werden muss. Danach folgt das Dollar-Zeichen oder das nationale Waehrungszeichen, je nachdem, welches Zeichen auf Ihrer Tastatur vorhanden ist. Beide Zeichen

sind gleichwertig.

Zuletzt steht der Name des Feldes, in dem nach der Zeichenreihe gesucht werden soll.

Bild 21 zeigt einmal das Durchsuchen der Datei "kunden" nach Personen, in deren Name ein "m" vorkommt und zum anderen das Durchsuchen nach Namen, in denen ein grosses "M" enthalten ist, die also mit "M" beginnen.
Die beiden Befehle lauten

```
LIST FOR "m"$name       und
LIST FOR "M"$name
```

```
: LIST FOR "m"$name
 Satznr.   NAME      VORNAME   PLZ    ORT         STRASSE
       1   Lehmann   Kurt      8060   DRESDEN     Landstr. 15
       2   Schmidt   Gisela    8223   THARANDT    Bahnhofstr. 43
       8   Neumann   Inge      8222   RABENAU     Gartenstr. 4
      10   Schmidt   Heinz     8223   THARANDT    Waldweg 42
:
:
: LIST FOR "M"$name
 Satznr.    NAME      VORNAME   PLZ    ORT         STRASSE
       3    Moser     Otto      8212   FREITAL     Neugraben 8
       6    Miller    Peter     8212   FREITAL     Nelkenweg 23
       7    Meier     August    8122   RADEBEUL    Uferstr. 12
```

Bild 21 Demonstration des Teilzeichenreihen-Operators

Wenn Sie nicht zwischen Gross- und Kleinschreibung unterscheiden wollen, koennen Sie die Funktionen

```
UPPER()   oder
LOWER()
```

bei der Formulierung Ihres LIST-Befehls verwenden. Sie erhalten also alle 7 im Bild 18 gefundenen Datensaetze, wenn Sie einen der folgenden Befehle eingeben:

```
LIST FOR 'm'$LOWER(name)
LIST FOR 'M'$UPPER(name)
```

Die Funktion LOWER wandelt alle Grossbuchstaben in Kleinbuchstaben um.
Die Funktion UPPER verwandelt Kleinbuchstaben in Grossbuchstaben.
In unserem Fall wird also der Inhalt des Zeichenfeldes "name" fuer die Dauer des Durchsuchens mittels Befehl LIST vor dem Vergleich umgewandelt.
Bild 22 demonstriert dies.

```
: LIST FOR "m"$LOWER(name)
   Satznr.  NAME      VORNAME   PLZ    ORT       STRASSE
         1  Lehmann   Kurt      8060   DRESDEN   Landstr. 15
         2  Schmidt   Gisela    8223   THARANDT  Bahnhofstr. 43
         3  Moser     Otto      8212   FREITAL   Neugraben 8
         6  Miller    Peter     8212   FREITAL   Nelkenweg 23
         7  Meier     August    8122   RADEBEUL  Uferstr. 12
         8  Neumann   Inge      8222   RABENAU   Gartenstr. 4
        10  Schmidt   Heinz     8223   THARANDT  Waldweg 42
```

Bild 22 Anwenden des Teilzeichenreihen-Operators und der LOWER-Funktion

## Verknuepfen von Suchbedingungen

Will man mehrere vergleichende Bedingungen innerhalb eines Befehls formulieren, geschieht das Verknuepfen mit den drei logischen Grundfunktionen

```
.AND.  (entspricht "und", d. h. sowohl...als auch)
.OR.   (entspricht "oder", d. h. entweder...oder)
.NOT.  (entspricht "nicht")
```

Diese sind jeweils in Punkte einzuschliessen. Im gewoehnlichen Sprachgebrauch wird "oder" i.a. mit ausschliessender Bedeutung benutzt, d.h. "A oder B" bedeutet "entweder nur A oder nur B". Das logische ODER dagegen bedeutet "entweder nur A oder nur B oder sowohl A als auch B".
Zunaechst sehen Sie im Bild 23 eine "OR"-Verknuepfung. Aus der Datei "kunden" sollen Personen herausgesucht werden, deren Name entweder mit "M" beginnt oder deren Vorname Fred ist.

```
: LIST FOR ("M"$name .OR. vorname="Fred")
  Satznr.  NAME     VORNAME   PLZ    ORT       STRASSE
        3  Moser    Otto      8212   FREITAL   Neugraben 8
        6  Miller   Peter     8212   FREITAL   Nelkenweg 23
        7  Meier    August    8122   RADEBEUL  Uferstr. 12
       13  Tauber   Fred      8028   DRESDEN   Fliederstr. 42
```

Bild 23 Zwei mit "OR" verknuepfte Bedingungen

Bild 24 zeigt ein Beispiel fuer zwei mit "AND" verknuepfte Bedingungen. Die Namen der gewuenschten Personen der Datei "kunden" sollen mit "M" beginnen und diese Kunden sollen im Kreis Freital wohnen, d. h. die Postleitzahl muss mit den Ziffern 82 beginnen.

```
: LIST FOR ("M"$name .AND. plz = "82")
  Satznr. NAME      VORNAME   PLZ    ORT       STRASSE
        3  Moser    Otto      8212   FREITAL   Neugraben 8
        6  Miller   Peter     8212   FREITAL   Nelkenweg 23
```

Bild 24 Zwei mit "AND" verknuepfte Bedingungen

Eindeutigkeit von Suchbedingungen
---

Es ist moeglich, den Teilzeichenreihen-Operator zur Funktion "SUBSTR()" (Substring) zu erweitern, so dass die Position des Suchbegriffs innerhalb des Feldes genau spezifiziert werden kann:

```
LIST FOR <ausdrC0>=SUBSTR(<ausdrC>,<ausdrN1>[,<ausdrN2>)]
```

Wichtig ist das Gleichheitszeichen nach dem Suchbegriff. Unter <ausdrC> ist in diesem Beispiel der Feldname zu verstehen (vgl. hierzu Abschnitt 9.1.)
<ausdrN1> ist diejenige Zeichenposition im Feld, von der ab verglichen werden soll, <ausdrN2> ist ebenfalls ein Zahlenwert, der die Anzahl der zu vergleichenden Zeichen angibt.

Bild 25 zeigt, wie dieser Befehl gehandhabt wird. (Im Prinzip erhalten Sie in diesem Beispiel die gleichen Resultate, die Sie beispielsweise auch mit LIST FOR plz = "80" erhielten.) Wir moechten Sie lediglich darauf hinweisen, dass Sie bei Nichtpositionierung des Suchbegriffs "80", also mit dem Befehl

```
LIST FOR "80"$plz
```

alle Personen faenden, die zufaellig innerhalb der vierstelligen Postleitzahl die Ziffernfolge "80" fuehren. Die Suchbedingung waere so also nicht eindeutig formuliert, denn Sie wollten nur die Kunden aufgelistet haben, die im Stadtgebiet Dresden wohnen, d. h. deren Postleitzahl mit 80 beginnt.

```
: LIST FOR "80"=SUBSTR(plz,1,2)
 Satznr.  NAME      VORNAME   PLZ    ORT        STRASSE
       1  Lehmann   Kurt      8060   DRESDEN    Landstr. 15
       5  Schulze   Ernst     8060   DRESDEN    Hauptstr. 61
      12  Heinze    Max       8060   DRESDEN    Rosenstr. 78
      13  Tauber    Fred      8028   DRESDEN    Fliederstr. 42
```

Bild 25 Datenbankdatei "kunden", mit positioniertem Suchbegriff durchsucht

3.2.3. Das Ausblenden von Datensaetzen
---

REDABAS-4 gestattet das Ausblenden von Datensaetzen ueber einen REDABAS-4-Befehl hinaus. Haben Sie bisher mit

```
LIST FOR <bedingung>
```

Datensaetze von der weiteren Verarbeitung ausgeschaltet, weil sie der <bedingung> nicht entsprachen, so galt das nur fuer den Zeitraum dieser einmaligen Auswertung mit dem LIST-Befehl.

Mit

```
SET FILTER TO <bedingung>
```

werden Datensaetze, die der <bedingung> nicht genuegen, so lange von einer Weiterverarbeitung ausgeklammert, bis entweder ein

- SET FILTER TO (ohne Bedingung) gesetzt wird
- ein erneuter Befehl SET FILTER TO <bedingung> mit neu definierter <bedingung> gegeben wird oder
- die Datei, auf die der FILTER-Befehl wirken soll, nicht mehr geoeffnet ist.

Mit diesem Befehl wird es moeglich, REDABAS-4-Befehle nur auf bestimmte Datensaetze wirken zu lassen, ohne dass bei jedem Befehl eine FOR-Bedingung gesetzt werden muss. Alle Operationen nach dem SET FILTER TO-Befehl beziehen sich also nur auf einen Teil der urspruenglich eroeffneten Datenbankdatei. Weitere Moeglichkeiten des SET FILTER TO-Befehls finden Sie in den Abschnitten 3.9. und 9.3.3.

Im Beispiel von Bild 26 sollen bestimmte Auswertungen nur fuer Kunden der Stadt Dresden durchgefuehrt werden. Der einmalig am Beginn gegebene SET FILTER-Befehl blendet fuer alle drei LIST-Befehle Personen mit anderen Wohnorten aus. (Den abschliessenden SET FILTER TO-Befehl sollten Sie unbedingt eingeben, wenn Sie anschliessend mit der vollstaendigen Datenbankdatei "kunden" weiterarbeiten!)

```
: SET FILTER TO ort="DRESDEN"
: LIST
 Satznr.  NAME      VORNAME PLZ   ORT       STRASSE
       1  Lehmann   Kurt    8060  DRESDEN   Landstr. 15
       5  Schulze   Ernst   8060  DRESDEN   Hauptstr. 61
      12  Heinze    Max     8060  DRESDEN   Rosenstr. 78
      13  Tauber    Fred    8028  DRESDEN   Fliederstr. 42

: LIST name,strasse
 Satznr.  name      strasse
       1  Lehmann   Landstr. 15
       5  Schulze   Hauptstr. 61
      12  Heinze    Rosenstr. 78
      13  Tauber    Fliederstr. 42

: LIST FOR "n"$name
 Satznr.  NAME      VORNAME   PLZ   ORT       STRASSE
       1  Lehmann   Kurt      8060  DRESDEN   Landstr. 15
      12  Heinze    Max       8060  DRESDEN   Rosenstr. 78

: SET FILTER TO
```

Bild 26 Das Ausblenden von Datensaetzen

### 3.2.4. Bedingte Suche einzelner Datensaetze

REDABAS-4 bietet ausser dem ausfuehrlich besprochenen Befehl LIST weitere Varianten des Durchsuchens der Datenbankdatei, die nachfolgend erklaert werden sollen.

LIST bringt stets alle zutreffenden Datensaetze gemeinsam zur Anzeige. Will man die Saetze jedoch einzeln aufrufen, beispielsweise um sie spaeter zu bearbeiten, geschieht das mit dem Befehl

```
LOCATE FOR <bedingung>
```

Es wird nach dem ersten Satz gesucht, der die gestellte Bedingung erfuellt. Wird ein Satz gefunden, zeigt REDABAS-4 die Satznummer an. Der Satz selbst wird mit LOCATE nicht auf den Bildschirm gebracht. Wird die Anzeige des Satzes jedoch gewuenscht, geschieht das nach dem LOCATE-Befehl mit

```
DISPLAY
```

Der Datensatz erscheint jetzt in der gleichen Form, wie Sie es von LIST her kennen. Unabhaengig davon, ob Sie sich den Datensatz anzeigen lassen, koennen Sie mit

```
CONTINUE
```

die Suche fortsetzen und, wenn Sie es wuenschen, den naechsten gefundenen Satz wiederum mit DISPLAY anzeigen lassen.
Bild 27 demonstriert Ihnen dies.

```
: LOCATE FOR "Schmidt"$name
Satz =     2

: DISPLAY
 Satznr.  NAME      VORNAME    PLZ   ORT        STRASSE
       2  Schmidt   Gisela     8223  THARANDT   Bahnhofstr. 43

: CONTINUE
Satz =       10

: DISPLAY
 Satznr.  NAME      VORNAME    PLZ   ORT        STRASSE
      10  Schmidt   Heinz      8223  THARANDT   Waldweg 42
```

Bild 27 Beispiel fuer die Anwendung der Befehle LOCATE, DISPLAY, CONTINUE

Der Befehl LOCATE FOR wird bei spezifizierten Bedingungen genau wie der Befehl LIST FOR formuliert.

### 3.2.5. Bedingte Anzeige von Datensaetzen

Wir moechten Ihnen den Befehl DISPLAY nachfolgend etwas ausfuehrlicher erlaeutern, da er fuer die Arbeit mit REDABAS-4 sehr wichtig ist.

Enthaelt der Befehl DISPLAY keinerlei Zusatzangaben, bezieht er sich auf den aktuellen Satz. (LIST ohne Zusatzangaben bringt hingegen alle Saetze der Datei zur Anzeige.)

Ist eine FOR-Klausel vorhanden, werden saemtliche Datensaetze, die die Bedingung erfuellen, angezeigt.
Nach jeweils 20 Datensaetzen (bei Saetzen, deren Anzeige jeweils nur eine Bildschirmzeile belegt, sonst entsprechend weniger) gibt Ihnen REDABAS-4 Gelegenheit, diese zu betrachten, und erst nach Druecken irgendeiner Taste werden die naechsten 20 Datensaetze zur Anzeige gebracht. Hier liegt uebrigens ein wesentlicher Unterschied zum LIST-Befehl, der alle Informationen ohne Unterbrechung anzeigt.

DISPLAY ALL zeigt alle in der Datenbankdatei vorhandenen Datensaetze an. Nach jeweils maximal 20 Saetzen (wenn der Bildschirm gefuellt ist) wird eine Pause gemacht.

DISPLAY RECORD <ausdrN> zeigt den durch die Angabe der Satznummer spezifizierten Datensatz an. (<ausdrN> steht fuer numerischen Ausdruck, vgl. Abschnitt 9.1).

DISPLAY NEXT <ausdrN> listet so viele der nachfolgenden Datensaetze auf, wie unter <ausdrN> spezifiziert wurden. Begonnen wird mit dem aktuellen Satz.

Zwei Formen des DISPLAY-Befehls zeigt Bild 28.

```
: DISPLAY RECORD 5
 Satznr. NAME       VORNAME    PLZ   ORT        STRASSE
       5 Schulze    Ernst      8060  DRESDEN    Hauptstr. 61

: DISPLAY NEXT 4
 Satznr. NAME       VORNAME    PLZ   ORT        STRASSE
       5 Schulze    Ernst      8060  DRESDEN    Hauptstr. 61
       6 Miller     Peter      8212  FREITAL    Nelkenweg 23
       7 Meier      August     8122  RADEBEUL   Uferstr. 12
       8 Neumann    Inge       8222  RABENAU    Gartenstr. 4
```

Bild 28 Varianten des Befehls DISPLAY

### 3.2.6. Positionieren auf Datensaetze

Wird eine Datenbankdatei mittels USE eroeffnet, wird durch REDABAS-4 der sogenannte Datensatzzeiger stets auf die Datensatznummer 1 gestellt. Waehrend der Bearbeitung einer Datei steht dieser Datensatzzeiger dann auf dem jeweils aktivierten Datensatz. Will man von dort aus wieder an den Dateianfang zurueck, geschieht das mit dem Befehl

```
GO TOP
```

Der Befehl

```
GO BOTTOM
```

setzt den Datensatzzeiger auf den letzten Satz. Damit kann man sehr schnell auf die letzte Eintragung der Datei zugreifen.

```
GO <ausdrN>
```

positioniert den Datensatzzeiger auf jeden beliebigen Datensatz. Es genuegt auch, nur die Nummer des gewuenschten Datensatzes in der Befehlszeile anzugeben.

```
GO 6
: DISPLAY
 Satznr.  NAME     VORNAME    PLZ   ORT        STRASSE
       6  Miller   Peter      8212  FREITAL    Nelkenweg 23

: GO TOP
: DISPLAY
 Satznr.  NAME     VORNAME    PLZ   ORT        STRASSE
       1  Lehmann  Kurt       8060  DRESDEN    Landstr. 15

: GO BOTTOM
: DISPLAY
 Satznr.  NAME     VORNAME    PLZ   ORT        STRASSE
      13  Tauber   Fred       8028  DRESDEN    Fliederstr. 42

: 8
: DISPLAY
 Satznr.  NAME     VORNAME    PLZ   ORT        STRASSE
       8  Neumann  Inge       8222  RABENAU    Gartenstr. 4
```

Bild 29 Varianten des Befehls GO

### 3.2.7. Abfragen einzelner Datenfelder

Sowohl mit dem Befehl LIST als auch mit dem Befehl DISPLAY ist es moeglich, einzelne Felder eines Datensatzes fuer eine Anzeige zu spezifizieren (vgl. LIST Vorname,Name,Ort). In der Praxis wird es haeufig der Fall sein, dass nicht alle Felder eines Satzes fuer eine bestimmte Aussage gewuenscht werden bzw. wegen mangelnden Platzes auf dem Bildschirm nur ein Teil der Felder angezeigt werden kann.

Eine Moeglichkeit, gezielt einzelne Datenfelder eines Datensatzes abzufragen, ist die Eingabe eines Fragezeichens mit nachfolgendem Feldnamen, wenn der Datensatz bereits im Zugriff ist oder vorher aufgerufen wurde. Bild 30 demonstriert dies.

Typisch fuer die Anzeige mit dem ?-Befehl ist, dass der gefundene Satz ohne die Feldnamen (d. h. ohne "Ueberschrift") angezeigt wird.

```
   LOCATE FOR vorname = "Otto"
  Satz =       3
 : ? name
 Moser
 : ? plz, ort, strasse
 8212 FREITAL  Neugraben 8
```

Bild 30  Abfragen einzelner Felder mit dem Fragezeichen

Uebrigens koennen Sie das Fragezeichen auch nutzen, um kleine Rechnungen -aehnlich wie mit dem Taschenrechner- vorzunehmen. Sie geben nach dem Fragezeichen einfach die zu berechnenden Werte ein und erhalten in der naechsten Zeile das Ergebnis. REDABAS-4 gestattet die mathematischen Operationen Addition, Subtraktion, Multiplikation, Division, Potenzierung, Quadratwurzel, Exponentialfunktion und Logarithmus. Ausserdem gibt es eine Reihe weiterer mathematischer Funktionen, wie z.B. Runden (ROUND()), Absolutwertberechnung (ABS()). Vergleichen Sie hierzu Abschn. 9.2.

## 3.3. Aendern der Dateistruktur

In den vorangegangenen Abschnitten lernten Sie all jene Befehle kennen, mit denen Sie eine Datenbank aufbauen, pflegen und auswerten koennen.
In der praktischen Datenbankarbeit wird es aber haeufig vorkommen, dass sich auf Grund neuer Gesichtspunkte Aenderungen an vorhandenen Datenbestaenden ergeben. Das heisst, Datenfelder muessen an bestehende Dateien hinzugefuegt oder von diesen entfernt werden, konkret, die Struktur der Datei muss veraendert werden.

REDABAS-4 bietet als relationales Datenbankbetriebssystem fuer diesen Datenbankdienst weitestgehend Datenunabhaengigkeit, d.h. die Aenderung der Dateistruktur erfolgt relativ einfach, ohne vorhandene Anwendungsprogramme zu beruehren.
Zum Aendern der Struktur einer aktivierten Datenbankdatei dient der Befehl

```
MODIFY STRUCTURE
```

Dieser Befehl ermoeglicht das Hinzufuegen und Loeschen von Datenfeldern sowie das Veraendern der Spezifikation der Felder.
Wir moechten Ihnen die Vorgehensweise beim Aendern einer Dateistruktur wiederum an unserer Datei "kunden" erlaeutern.
Fuer die Arbeit in der Praxis stellte sich heraus, dass es zweckmaessig ist, an jeden Kunden zur eindeutigen Identifikation eine Kundennummer zu vergeben.
Wir moechten deshalb in die eingangs definierte Struktur (vergl. Bild 5) ein Datenfeld namens "kunr" einfuegen.
Sie eroeffnen die Datei (falls nicht bereits eroeffnet) mit

```
USE kunden
```

und geben ein

```
MODIFY STRUCTURE
```

Daraufhin wird der Bildschirm geloescht, und die Struktur der Datei "kunden" wird - wie Sie sie bereits vom Bild 5 kennen - angezeigt.
Innerhalb dieser Maske koennen Sie mit den schon vom EDIT-Befehl bekannten Tastenkombinationen (vergl. Abschnitt 3.1.3.) den Cursor beliebig bewegen und Eintragungen veraendern.
Beachten Sie bitte gegenueber EDIT folgende Veraenderungen:

<CTRL-U> Loeschen der Felddefinition

<CTRL-N> Einfuegen einer Leerzeile fuer die Eintragung eines neuen Datenfeldes

Wir moechten jetzt in unserem Beispiel fortfahren und in die Datei "kunden" zusaetzlich das Feld "Kundennummer" aufnehmen.

Bewegen Sie den Cursor in die erste Zeile und druecken Sie <CTRL-N>. So koennen Sie noch vor dem Namen die Kundennummer einfuegen. Der Name des neuen Feldes soll "kunr" sein, der Typ "Zeichen", und die Laenge soll drei Stellen betragen.

Sie tragen also folgende Informationen in das neue Feld 1 ein:

```
  Feldname        Typ             Laenge          Dez
1 KUNR            ZEICHEN           3
```

Nach dem Betaetigen von <CTRL-End> oder <CTRL-W> und abschliessendem Bestaetigen mit <ET> wird die neue Struktur gespeichert, und Sie kehren zur Befehlseingabe von REDABAS-4 zurueck. Moechten Sie sich ueberzeugen, dass die Aenderung ordnungsgemaess erfolgt ist, geben Sie DISPLAY STRUCTURE oder LIST STRUCTURE ein. Sie koennen auch mit <CTRL-P> den Drucker zuschalten, um die neue Struktur zu protokollieren. Bild 31 zeigt die neue Struktur, Bild 32 die geaenderte Datei.

```
   DISPLAY STRUCTURE
 Datenbankdateistruktur :   A:kunden.dbd
 Anzahl der Datensaetze :        13
 Letztes Aenderungsdatum:   02.09.89
 Feld   Feldname   Typ        Laenge   Dez
    1   KUNR       Zeichen      3
    2   NAME       Zeichen     10
    3   VORNAME    Zeichen      8
    4   PLZ        Zeichen      4
    5   ORT        Zeichen      8
    6   STRASSE    Zeichen     15
 * Gesamt *                    49
```

Bild 31 Neue Struktur der Datenbankdatei "kunden"

```
 :LIST
 Satznr. KUNR NAME       VORNAME PLZ  ORT       STRASSE
       1      Lehmann    Kurt    8060 DRESDEN   Landstr. 15
       2      Schmidt    Gisela  8223 THARANDT  Bahnhofstr. 43
       3      Moser      Otto    8212 FREITAL   Neugraben 8
       4      Weber      Franz   8223 THARANDT  Heideweg 66
       5      Schulze    Ernst   8060 DRESDEN   Hauptstr. 61
       6      Miller     Peter   8212 FREITAL   Nelkenweg 23
       7      Meier      August  8122 RADEBEUL  Uferstr. 12
       8      Neumann    Inge    8222 RABENAU   Gartenstr. 4
       9      Fleischer  Karl    8212 FREITAL   Am See 83
      10      Schmidt    Heinz   8223 THARANDT  Waldweg 42
      11      Lolle      Peter   8122 RADEBEUL  Goethestr. 13
      12      Heinze     Max     8060 DRESDEN   Rosenstr. 78
      13      Tauber     Fred    8028 DRESDEN   Fliederstr. 42
```

Bild 32 Geaenderte Datenbankdatei "kunden"

Bevor Sie die Kundennummern eintragen, moechten wir Ihnen noch sagen, wie Sie - falls Sie mit MODIFY STRUCTURE Aenderungen durchfuehren, die Sie eventuell wieder rueckgaengig machen moechten - die urspruengliche Datei wieder verfuegbar machen koennen.
REDABAS-4 sichert vor dem Modifizieren automatisch die alte Datei und vergibt dafuer den Dateityp .BAK. Diese Sicherungsdatei koennen Sie aufrufen, muessen sie aber, bevor Sie sie weiterverarbeiten, in eine Datei vom Typ .DBD umbenennen.

Die Kundennummern koennen Sie in die Datei eintragen, indem

Sie mit dem Befehl EDIT die Datensaetze einzeln aufrufen und ergaenzen. Wir moechten Ihnen aber einen Befehl vorstellen, mit dem Sie so viele Datensaetze gleichzeitig aufrufen und bearbeiten koennen, wie auf Ihrem Bildschirm Platz haben. Dieser maechtige Befehl heisst

BROWSE

Sie koennen mit BROWSE gewissermassen "in der Datei blaettern". Die Datensaetze sind zeilenweise angeordnet und koennen wunschgemaess mit den Cursortasten und den entsprechenden Tastenkombinationen bearbeitet werden.

Eine wesentliche Voraussetzung fuer die Anwendung des BROWSE-Befehls ist das Einstellen des Datensatzzeigers auf den Dateianfang. Dies geschieht mit

GO TOP

Um unsere Datei "kunden" nun mit der Kundennummer zu komplettieren, geben Sie bitte ein:

```
GO TOP
BROWSE
```

Auf dem Bildschirm werden Ihnen die einzelnen Datensaetze zeilenweise angezeigt. Sie positionieren den Cursor auf das Feld "kunr" des ersten Datensatzes und geben die Ziffern 048 ein. Bild 33 zeigt Ihnen die weiteren Kundennummern, die Sie bitte alle nacheinander eintragen.
Nachdem Sie die letzte Kundennummer eingetragen haben (d.h. Datensatz Nr. 13 bearbeitet wurde - siehe Statuszeile) und versucht haben, auf den naechsten Satz einzustellen, fragt Sie REDABAS-4:

Neue Saetze hinzufuegen? (J/N)

Wir verneinen und schliessen die Bearbeitung mit <CTRL-W> oder <CTRL-End> ab und beenden damit den Befehl BROWSE. (Normalerweise haetten Sie schon nach Eingabe der letzten Kundennummer abschliessen koennen.)

| KUNR | NAME------ | VORNAME- | PLZ- | ORT----- | STRASSE-------- |
|---|---|---|---|---|---|
| 048 | Lehmann | Kurt | 8060 | DRESDEN | Landstr. 15 |
| 123 | Schmidt | Gisela | 8223 | THARANDT | Bahnhofstr. 43 |
| 645 | Moser | Otto | 8212 | FREITAL | Neugraben 8 |
| 850 | Weber | Franz | 8223 | THARANDT | Heideweg 66 |
| 267 | Schulze | Ernst | 8060 | DRESDEN | Nebenstr. 61 |
| 678 | Miller | Peter | 8212 | FREITAL | Nelkenweg 23 |
| 421 | Meier | August | 8122 | RADEBEUL | Uferstr. 12 |
| 789 | Neumann | Inge | 8222 | RABENAU | Gartenstr. 4 |
| 750 | Fleischer | Karl | 8212 | FREITAL | Am See 83 |
| 366 | Schmidt | Heinz | 8223 | THARANDT | Waldweg 42 |
| 910 | Lolle | Peter | 8122 | RADEBEUL | Goethestr. 13 |
| 083 | Heinze | Max | 8060 | DRESDEN | Rosenstr. 78 |
| 999 | Tauber | Fred | 8028 | DRESDEN | Fliederstr. 42 |

Bild 33 Durch den Befehl BROWSE mit der Kundennummer komplettierte Datenbankdatei "kunden"

Der BROWSE-Befehl - den Sie hier in seiner allgemeinsten Form kennenlernten - laesst in der Anwendung viele Varianten zu. So koennen Sie beispielweise mit

```
BROWSE FIELDS name,kunr
```

nur diese beiden Felder anzeigen lassen und die Eintragungen vornehmen. (Der Cursor springt dann automatisch ins naechste Feld.)
BROWSE bietet Ihnen ausserdem die Moeglichkeit, ein Menue der im BROWSE-Modus verfuegbaren Optionen durch Betaetigen von <CTRL-Home> aufzurufen:

Ende Anfang Fixieren Satz Nr. Grenze

Mit den Cursortasten waehlen Sie die gewuenschte Option, die Sie mit <ET> bestaetigen. Auf der unteren Bildschirmzeile erhalten Sie entsprechende Erklaerungen. So koennen Sie muehelos auf Datensaetze positionieren sowie Felder fuer das Editieren auswaehlen.

## 3.4. Sortieren und Indizieren

### 3.4.1. Sortieren

Da Daten im allgemeinen in keiner bestimmten Reihenfolge erfasst werden, wird es fuer auszudruckende Berichte, das Verdichten oder Auswerten von Datenbankdateien oder fuer besonders schnellen Zugriff erforderlich sein, Daten durch Sortieren und Indizieren neu zu organisieren.

REDABAS-4 bietet fuer das Sortieren einer Datenbankdatei den Befehl

```
SORT TO <db-datei> ON <feldfolge>
```

Die aktivierte Datenbankdatei wird nach einem oder mehreren zu benennenden Datenfeldern sortiert. Normalerweise erfolgt die Sortierung aufsteigend. Geben Sie ein /D hinter dem Feldnamen an, wird absteigend sortiert.
Die sortierte Datenbankdatei wird in die mit "db-datei" spezifizierte Zieldatei abgespeichert, deren Name sich von der Ursprungsdatei unbedingt unterscheiden muss.

Im folgenden soll die Datenbankdatei "kunden" nach aufsteigender Kundennummer sortiert werden. Nach dem Sortieren werden die Datensaetze in der temporaeren Datei "temp" zwischengespeichert, die temporaere Datei wird eroeffnet und von dieser sortierten Datei wird eine Kopie unter dem urspruenglichen Namen kunden.dbd erzeugt.
Fuer das Kopieren einer aktivierten Datenbankdatei in eine andere dient der Befehl

```
COPY TO <datei>
```

Wenn Sie die folgende Befehlsfolge eingeben, erhalten Sie die in Bild 34 dargestellte, nach der Kundennummer sortierte Datei "kunden".

```
USE kunden
SORT TO temp ON kunr
USE temp
COPY to kunden
USE kunden
LIST
```

Bild 34 zeigt den Ablauf von Befehlen, die Systemnachrichten und als Ergebnis die nach der Kundennummer sortierte Datenbankdatei "kunden".

```
: USE kunden
: SORT TO temp ON kunr
  100 % sortiert          13 Saetze sortiert
: USE temp
: COPY TO kunden
kunden.dbd bereits vorhanden, Ueberschreiben erwuenscht?
(J/N) Ja
    13 Saetze kopiert
: USE kunden
: LIST
 Satznr.KUNR NAME      VORNAME PLZ  ORT       STRASSE
       1 048  Lehmann   Kurt    8060 DRESDEN   Landstr. 15
       2 083  Heinze    Max     8060 DRESDEN   Rosenstr. 78
       3 123  Schmidt   Gisela  8223 THARANDT  Bahnhofstr. 43
       4 267  Schulze   Ernst   8060 DRESDEN   Nebenstr. 61
       5 366  Schmidt   Heinz   8223 THARANDT  Waldweg 42
       6 421  Meier     August  8122 RADEBEUL  Uferstr. 12
       7 645  Moser     Otto    8212 FREITAL   Neugraben 8
       8 678  Miller    Peter   8212 FREITAL   Nelkenweg 23
       9 750  Fleischer Karl    8212 FREITAL   Am See 83
      10 789  Neumann   Inge    8222 RABENAU   Gartenstr. 4
      11 850  Weber     Franz   8223 THARANDT  Heideweg 66
      12 910  Lolle     Peter   8122 RADEBEUL  Goethestr. 13
      13 999  Tauber    Fred    8028 DRESDEN   Fliederstr. 42
```

Bild 34 Datenbankdatei "kunden", sortiert nach der Kundennummer

Wir moechten Sie noch darauf hinweisen, dass das Zwischenspeichern in eine temporaere Datei unbedingt erforderlich ist. Eine Datei kann nicht in sich selbst sortiert werden. Durch das Zwischenspeichern bleibt die Ursprungsdatei so lange erhalten, bis Sie davon ueberzeugt sind, dass der Sortierablauf ordnungsgemaess ausgefuehrt wurde.

Eine neu sortierte Datei bewirkt auch eine neue Zuordnung zwischen den Datensaetzen und deren laufenden Satznummern (im Beispiel ist der ehemalige Satz Nr. 3 nach dem Sortieren z. B. Satz Nr. 7 geworden). Die Satznumerierung geschieht intern durch REDABAS-4 und ist unabhaengig vom Inhalt der Datensaetze. Dies ist besonders dann zu beachten, wenn Sie Befehle ausfuehren, die auf die Satznummern Bezug nehmen (wie z.B. EDIT).

Generell ist es moeglich, jede beliebige Sortierung einer Datei mittels temporaerer Datei vorzunehmen. Eine solche zweite, neben der "Stammdatei" laufende Datei sollte jedoch nur fuer

eine einmalige Auswertung benutzt werden, um unnoetige und gefaehrliche Redundanzen in der Datenbank zu vermeiden. Man denke beispielsweise an das Problem der Aktualisierung.

Sie koennen mit REDABAS-4 nach maximal 10 Feldern sortieren. Die Reihenfolge der Sortierbegriffe bestimmt die Sortierprioritaet.

In einem zweiten Beispiel moechten wir die Datei "kunden" nach zwei Kriterien sortieren. Es soll nach den Wohnorten der Kunden (absteigend) und innerhalb der Orte nach den Namen (aufsteigend) sortiert werden.
Der entsprechende SORT-Befehl lautet:

```
SORT TO temp ON ort/D,name
```

Bild 35 zeigt die Befehlsfolge mit REDABAS-4-Nachrichten und die neu sortierte Datei.

```
: USE kunden
: SORT TO temp ON ort/D,name
temp.dbd bereits vorhanden,  Ueberschreiben erwuenscht?
(J/N) Ja
     100 % sortiert              13 Saetze sortiert
: USE temp
: COPY TO kunden
kunden.dbd bereits vorhanden, Ueberschreiben erwuenscht?
(J/N) Ja
     13 Saetze kopiert
: USE kunden
: LIST
 Satznr. KUNR NAME       VORNAME PLZ  ORT       STRASSE
       1 123  Schmidt    Gisela  8223 THARANDT  Bahnhofstr. 43
       2 366  Schmidt    Heinz   8223 THARANDT  Waldweg 42
       3 850  Weber      Franz   8223 THARANDT  Heideweg 66
       4 910  Lolle      Peter   8122 RADEBEUL  Goethestr. 13
       5 421  Meier      August  8122 RADEBEUL  Uferstr. 12
       6 789  Neumann    Inge    8222 RABENAU   Gartenstr. 4
       7 750  Fleischer  Karl    8212 FREITAL   Am See 83
       8 678  Miller     Peter   8212 FREITAL   Nelkenweg 23
       9 645  Moser      Otto    8212 FREITAL   Neugraben 8
      10 083  Heinze     Max     8060 DRESDEN   Rosenstr. 78
      11 048  Lehmann    Kurt    8060 DRESDEN   Landstr. 15
      12 267  Schulze    Ernst   8060 DRESDEN   Hauptstr. 61
      13 999  Tauber     Fred    8028 DRESDEN   Fliederstr. 42
```

Bild 35 Datenbankdatei "kunden", sortiert nach Ort und Name

Im Anschluss wird die temporaere Datei mit

```
ERASE temp.dbd
```

wieder geloescht.

### 3.4.2. Indizieren

Neben dem Sortieren gibt es noch eine zweite Moeglichkeit, Datenbestaende in einer Sortierordnung zu verwalten, das Indizieren.
Der Unterschied zwischen Sortieren und Indizieren besteht im folgenden:
Waehrend einer Sortierung werden die Datensaetze in der Datei tatsaechlich in ihrer Gesamtheit neu geordnet. Es wird eine sortierte Kopie der Datenbankdatei erstellt. Da die Menge der Datensaetze im allgemeinen umfangreich ist, kann es mehrere Sekunden dauern, einen bestimmten Datensatz wiederaufzufinden.
Bei der Indizierung hingegen werden die Datensaetze in der Datei auf ihren Plaetzen belassen, und es wird eine Hilfsdatei (Indexdatei) erzeugt, die pro Satz nur ein einziges Datenfeld enthaelt (den sogenannten Schluessel), nach dem die Datei sortiert ist. Ausserdem enthaelt sie zu jedem Schluesselwert einen "Zeiger" auf die Position des Datensatzes in der Datei. Dies hat den Vorteil, dass man, unabhaengig von der Groesse der Datei, sehr schnell den gesuchten Schluessel findet und man durch den Zeiger sofort den entsprechenden Datensatz in der Stammdatei wiederauffindet, ohne die gesamte Datei durchsuchen zu muessen.

Zu einer Datenbankdatei koennen auf diese Weise beliebig viele Indexdateien erstellt werden.

Der Index-Befehl selbst lautet:

```
INDEX ON <ausdr> TO <indexdatei>
```

Der Name der "indexdatei" ist frei waehlbar. REDABAS-4 versieht den Namen einer Indexdatei intern mit dem Zusatz .IDX (im Unterschied zur Stammdatei, die vom Typ .DBD ist).

Um das Indizieren zu demonstrieren, erstellen wir zur Datei "kunden" eine Indexdatei "kunort", um eine Sortierung der Kunden nach Orten, d.h. eigentlich Postleitzahlen vorzunehmen. Sie geben ein:

```
USE kunden
INDEX ON plz TO kunort
```

Es erscheint die Systemnachricht:

```
100% indiziert        13 Saetze indiziert
```

Das Eroeffnen der Indexdatei erfolgt durch gemeinsamen Aufruf von Stammdatei und Indexdatei:

```
USE kunden INDEX kunort
```

Damit ist diese Indexdatei wie jede andere Datei verfuegbar. Bild 36 zeigt die nach den Orten indizierte Datei "kunden". Durch den LIST-Befehl wird die nach Postleitzahlen indizierte Datei entsprechend umgeordnet ausgegeben. Die Stammdatei "kunden" bleibt weiterhin wie in Bild 35 gezeigt, sortiert. Man erkennt dies auch daran, dass die Satznummern noch denselben Datensaetzen wie vor der Indizierung zugeordnet sind.

Die Zuordnung der Satznummern ist wiederum unbedingt bei der

Anwendung von Befehlen zu beachten, die auf diese Nummern Bezug nehmen. Das gilt beispielsweise bei Gebrauch der Befehle GO TOP oder GO BOTTOM.

GO TOP zeigt in der Datei "kunden" auf Datensatz Nr. 1, in der Indexdatei auf Satz Nr. 13, GO BOTTOM in der Datei "kunden" auf Satz Nr. 13, in der Indexdatei auf Satz Nr. 3.

```
: USE kunden INDEX kunort
: LIST
 Satznr. KUNR NAME       VORNAME PLZ  ORT       STRASSE
      13 999  Tauber     Fred    8028 DRESDEN   Fliederstr. 42
      10 083  Heinze     Max     8060 DRESDEN   Rosenstr. 78
      11 048  Lehmann    Kurt    8060 DRESDEN   Landstr. 15
      12 267  Schulze    Ernst   8060 DRESDEN   Hauptstr. 61
       4 910  Lolle      Peter   8122 RADEBEUL  Goethestr. 13
       5 421  Meier      August  8122 RADEBEUL  Uferstr. 12
       7 750  Fleischer  Karl    8212 FREITAL   Am See 83
       8 678  Miller     Peter   8212 FREITAL   Nelkenweg 23
       9 645  Moser      Otto    8212 FREITAL   Neugraben 8
       6 789  Neumann    Inge    8222 RABENAU   Gartenstr. 4
       1 123  Schmidt    Gisela  8223 THARANDT  Bahnhofstr. 43
       2 366  Schmidt    Heinz   8223 THARANDT  Waldweg 42
       3 850  Weber      Franz   8223 THARANDT  Heideweg 66
```

Bild 36 Datenbankdatei "kunden", indiziert nach der Postleitzahl

### 3.4.3. Wiederauffinden mittels Indexdateien

Legen Sie jetzt noch eine zweite Indexdatei an, indem Sie die Stammdatei nach Kundennummern indizieren. Sie soll den Namen "kunrkun" tragen. Geben Sie die Befehlsfolge ein:

```
USE kunden
INDEX ON kunr TO kunrkun
```

Durch Einfuehren dieser Datei wollen wir Sie mit dem Befehl

```
SEEK <ausdr>
```

vertraut machen.
Mit dem SEEK-Befehl laesst sich eine Indexdatei sehr schnell nach einem Suchbegriff, der Inhalt des bei <ausdr> spezifizierten Schluessels ist, abfragen. Die Suche ist deshalb so schnell, weil pro Datensatz nur ein Datenfeld (naemlich das indizierte) vorhanden ist und nicht die ganze Datei durchsucht werden muss. (Die typische Zeit fuer die Ausfuehrung eines SEEK-Befehls betraegt ein bis zwei Sekunden.)

Die Suche mit dem SEEK-Befehl wird beim ersten Datensatz beendet, der im indizierten Feld den <ausdr> enthaelt. Der gefundene Satz kann anschliessend bearbeitet werden.

Moechten Sie beispielsweise bestimmte Aussagen ueber den Kunden mit der Kundennummer 678, so geben Sie ein:

```
SEEK "678"
```

(Da die Kundennummer vom Typ "Zeichen" definiert wurde, muss sie auch entsprechend gekennzeichnet werden.)
Der Datensatz ist nach Erscheinen des Doppelpunktes im Zugriff, und Sie koennen Anfragen stellen, wie z.B.

```
? name,ort
```

Sie erhalten als Antwort

```
Miller   FREITAL
```

Sie brauchen den Feldinhalt, nach dem Sie suchen moechten, nicht direkt anzugeben. Er kann auch in einer sogenannten Speichervariablen stehen. (Ueber Speichervariablen erfahren Sie im Abschnitt 3.5.4. Naeheres.)
Wenn Sie die Variante des SEEK-Befehls

```
SEEK <speichvar>
```

nutzen, wird nach dem I n h a l t der Speichervariablen, nicht nach dem Vorhandensein des Namens von <speichvar> gesucht.

Ein zweiter Befehl fuer das Wiederauffinden mittels Indexdatei ist

```
FIND <zeichenreihe>/<zahl>
```

An der Syntax sehen Sie schon, dass der FIND-Befehl nicht so universell wie der SEEK-Befehl eingesetzt werden kann.
Je nachdem, worauf die Suche in der indizierten Datei gerichtet ist, wird FIND oder das universeller einsetzbare SEEK angewandt.

SEEK und FIND durchsuchen die Datei stets vom Dateianfang an und enden beim ersten gefundenen Satz. Sie brauchen also nicht mit GO TOP an den Dateianfang zu gehen. Dies unterscheidet beide Befehle von LOCATE (bei dem auch Dateiabschnitte durchsucht werden koennen). Allerdings duerfen Sie die Suche auch nicht mit CONTINUE fortsetzen.

## 3.5. Auswerten und Verdichten

Neben den bereits bekannten Befehlen fuer den Datenbankzugriff moechten wir in diesem Abschnitt weitere Moeglichkeiten vorstellen, um Dateien auszuwerten. Dazu gehoeren Summieren, Zaehlen, Mittelwertbildung, das Verwenden von Speichervariablen und das Verdichten.

Bevor wir die einzelnen Befehle beschreiben, soll an dieser Stelle zusaetzlich zur bereits existierenden Datenbankdatei "kunden" eine zweite Datenbankdatei, genannt "auftrag" eingefuehrt werden. Diese neue Datei enthaelt Angaben zur Auftragsnummer (aufnr) und Kundennummer (kunr), ueber entstehende Betraege (betrag), das Datum der Rechnungserstellung (datum) und eine Aussage darueber, ob die Rechnung bezahlt wurde oder nicht (beza).

Da Sie mit den Befehlen fuer den Aufbau einer Datenbankdatei bereits voll vertraut sind, zeigen wir Ihnen in den Bildern 37 und 38 die schon erstellte Struktur und die Datei selbst, ohne die Befehle im einzelnen zu erklaeren.

Datenbankdateistruktur : A:auftrag.dbd
Anzahl der Datensaetze : 11
Letztes Aenderungsdatum: 04.09.89

| Feld | Feldname | Typ | Laenge | Dez |
|---|---|---|---|---|
| 1 | AUFNR | Zeichen | 4 | |
| 2 | KUNR | Zeichen | 3 | |
| 3 | BETRAG | Numerisch | 8 | 2 |
| 4 | DATUM | Datum | 8 | |
| 5 | BEZA | Logisch | 1 | |
| * Gesamt * | | | 25 | |

Bild 37 Struktur der Datenbankdatei "auftrag"

| Satznr. | AUFNR | KUNR | BETRAG | DATUM | BEZA |
|---|---|---|---|---|---|
| 1 | 0123 | 421 | 15.00 | 23.01.89 | .T. |
| 2 | 0310 | 910 | 38.38 | 02.02.89 | .F. |
| 3 | 0534 | 678 | 293.85 | 15.02.89 | .T. |
| 4 | 1230 | 910 | 34.64 | 15.02.89 | .T. |
| 5 | 0211 | 789 | 38.18 | 23.02.89 | .T. |
| 6 | 2442 | 123 | 284.81 | 27.02.89 | .F. |
| 7 | 2676 | 048 | 0.00 | . . | .F. |
| 8 | 8423 | 421 | 0.00 | . . | .F. |
| 9 | 4567 | 789 | 0.00 | . . | .F. |
| 10 | 1234 | 850 | 35.54 | 27.02.89 | .T. |
| 11 | 6677 | 750 | 1420.00 | 17.02.89 | .F. |

Bild 38 Datenbankdatei "auftrag"

### 3.5.1. Summenbildung

Wir moechten im folgenden den zu erwartenden Gesamtumsatz, den Umsatz fuer einen ausgewaehlten Monat und die Summe der Betraege der noch nicht bezahlten Auftraege ermitteln.

Fuer diese Summenbildung ueber die gesamte Datei steht der Befehl

```
SUM <ausdrNfolge> FOR <bedingung>
```

zur Verfuegung. Es werden die Inhalte von numerischen Feldern bzw. damit gebildete Ausdruecke einer Datei summiert. Dabei ist es moeglich, bestimmte Bedingungen vorzugeben.

Zunaechst summieren wir die Betraege der Datei "auftrag", um den zu erwartenden Gesamtumsatz zu erhalten. Nachdem die Datei eroeffnet wurde, geben Sie ein

```
SUM betrag
```

REDABAS-4 informiert Sie

```
11 Saetze summiert
betrag
2160.40
```

Es waere auch moeglich, mehrere numerische Felder zu summieren, die Feldnamen muessen hintereinander angegeben werden, untereinander durch Kommas getrennt.

Nicht bezahlte Auftraege werden durch Formulieren der Bedingung "NOT" fuer das logische Feld "beza" ermittelt.

```
SUM betrag FOR .NOT. beza
6 Saetze summiert
betrag
1743.19
```

Es ist auch moeglich, das Ergebnis einer Summierung in einen separaten Speicher abzulegen und bei Bedarf wieder aufzurufen.

REDABAS-4 ermoeglicht das Definieren von max. 256 sogenannten "Speichervariablen", die vorwiegend Zahlenwerte (z. B. das Ergebnis einer Rechnung) oder alphanumerischen Text beinhalten koennen. Diese Speichervariablen werden einzeln durch Vergabe eines Namens definiert und adressiert.

Unter der Ueberschrift "Verwendung von Speichervariablen" (Abschnitt 3.5.4.) werden wir Sie noch detaillierter ueber diese Moeglichkeit von REDABAS-4 informieren (vergleichen Sie hierzu auch Abschnitt 8.2.).

Um das Ergebnis einer Summierung zu speichern, formulieren wir den Befehl wie folgt

```
SUM <ausdrNfolge> TO <speichvarNfolge>
```

Der Name einer oder mehrerer Speichervariablen ist frei waehlbar. Es ist jedoch sinnvoll, einerseits einen Bezug zum Feldnamen zu schaffen, andererseits von diesem beispielsweise durch Voranstellen eines "S" (abgeleitet von Speichervariable) abzuweichen. Die folgenden beiden Anweisungen demonstrieren das Speichern

```
SUM betrag TO Sbetrag
     11 Saetze summiert
     betrag
   2160.40

SUM betrag FOR .NOT. beza TO Soffen
      6 Saetze summiert
     betrag
   1743.19
```

Die Ergebnisse der Summierung werden sowohl auf dem Bildschirm angezeigt als auch gleichzeitig fuer die Dauer einer REDABAS-4 Sitzung in die entsprechenden Speichervariablen abgelegt. Von diesen Speichervariablen aus koennen sie ueber ihren Namen jederzeit wieder aufgerufen und weiterverarbeitet werden. Da die Speichervariablen nur im Arbeitsspeicher aufbewahrt werden (nicht auf der Diskette), muessen sie, will man sie ueber den aktuellen REDABAS-4-Lauf hinaus erhalten, mit SAVE fuer spaetere Programmlaeufe gesichert werden. RELEASE loescht die Speichervariablen wieder.

### 3.5.2. Zaehlen von Datensaetzen

Der Befehl

```
COUNT
```

zaehlt die Saetze einer Datenbankdatei. Es gelten wiederum die beim Summieren beschriebenen Moeglichkeiten, Bedingungen zu formulieren und Speichervariable zu nutzen.
Formulieren Sie folgende Befehle

```
COUNT
  11 Saetze

COUNT FOR beza
   5 Saetze

COUNT FOR betrag > 0
   8 Saetze

COUNT FOR betrag > 0 TO Sauftr
   8 Saetze
```

### 3.5.3. Mittelwertbildung

Mit dem Befehl

```
AVERAGE <ausdrNfolge> FOR <bedingung>
```

kann man den arithmetischen Mittelwert der unter <ausdrNfolge> spezifizierten numerischen Felder bzw. damit gebildeten Ausdruecke einer Datenbankdatei bilden. Sollen nicht alle Datensaetze der Datei fuer die Mittelwertbildung herangezogen werden, kann man den Bereich eingrenzen und Bedingungen formulieren. Das Ergebnis der Mittelwertbildung kann wieder in eine oder mehrere Speichervariablen abgelegt werden.
Im Beispiel soll der Mittelwert der bezahlten Betraege der Datei "auftrag" errechnet werden:

```
AVERAGE betrag FOR beza
    5 Saetze gemittelt
betrag
 83.44
```

### 3.5.4. Verwendung von Speichervariablen

Die Speichervariablen (vgl. Abschn. 3.5.1.) koennen auch direkt ueber die Tastatur Ihres Computers eingegeben werden. REDABAS-4 stellt dafuer mehrere Befehle zur Verfuegung. Die meisten stellen wir im Kapitel 4. "Programmieren" vor. Hier sei zunaechst auf den Befehl

```
STORE <ausdr> TO <speichvarfolge>
```

oder als Alternative verkuerzt (fuer das Anlegen e.i n e r Speichervariablen)

```
<speichvar> = <ausdr>
```

verwiesen. Dieser Befehl dient dazu, erlaeuternden Text, berechnete Werte oder Konstanten, die haeufig gebraucht werden, abzuspeichern, zu verarbeiten und wieder auszugeben. Der Wert von "ausdr" kann vom Datentyp Numerisch, Zeichen, Logisch oder Datum sein und wird in die zu benennende "speichvar" eingetragen. REDABAS-4 vergibt den Datentyp automatisch. (Beim Datentyp Datum sind allerdings einige Besonderheiten zu beachten, die Sie spaeter kennenlernen.)
Es soll die Summe aller unbezahlten Auftraege ausgegeben werden und mit einer entsprechenden Ausschrift versehen werden.
Sie formulieren

```
STORE "Summe aller unbezahlten Auftraege" TO Stext
(oder Stext="Summe aller unbezahlten Auftraege")
? Stext,Soffen
```

und erhalten folgende Ausschrift auf dem Bildschirm

```
Summe aller unbezahlten  Auftraege      1743.19
```

Es ist auch moeglich, Zahlenwerte als Konstanten zu speichern und wenn erforderlich, in die Rechnung einzubeziehen:

```
STORE 100  TO Sprozent    (oder Sprozent=100)
? Soffen/Sbetrag*Sprozent
```

Als Ergebnis erscheint: 80.69

Will man sich ueber Namen, Gueltigkeitsbereich, Datentyp und Inhalt aller definierten Speichervariablen informieren, geschieht das mit dem Befehl

```
DISPLAY MEMORY
```

Zusaetzlich wird die Anzahl aller definierten Speichervariablen und die Anzahl noch verfuebarer Speichervariablen ausgegeben.

Bild 39 zeigt im Ueberblick alle bis zum jetzigen Zeitpunkt erstellten Speichervariablen.

```
: DISPLAY MEMORY
SBETRAG          pub   N    2160.40   (      2160.40000000)
SOFFEN           pub   N    1743.19   (      1743.19000000)
SAUFTR           pub   N       8     (         8.00000000)
STEXT            pub   C  "Summe aller unbezahlten
                            Auftraege"
SPROZENT         pub   N     100     (       100.00000000)
    5 Speichervariable def.,          71 Bytes benutzt
  251 Speichervariable frei,        5929 Bytes frei
:
```

Bild 39 Uebersicht der Spechervariablen

Die Spalten der Tabelle geben folgende Informationen zu den einzelnen Speichervariablen:

- Name

- Gueltigkeitsbereich
  pub steht fuer "public" und weist auf den Status der Speichervariablen. Waehrend die sogenannten PUBLIC-Speichervariablen g l o b a l zur Verfuegung stehen und nicht geschuetzt sind, sind im Gegensatz dazu die PRIVATE-Speichervariablen nur l o k a l (d. h. in einem Programm) verfuegbar. Sie brauchen sich jetzt noch nicht dafuer zu interessieren, erst beim Programmieren mit REDABAS-4 muessen Sie auf den Gueltigkeitsbereich von Speichervariablen achten.

- Datentyp
  REDABAS-4 gestattet vier Datentypen: Zeichen (C), Numerisch (N), Logisch (L), Datum (D).

- Inhalt
  Bei numerischen Speichervariablen sehen Sie zwei Darstellungen unterschiedlicher Genauigkeit.

Vergleichen Sie bitte hierzu Abschnitt 8.2.

### 3.5.5. Verdichten von Dateien

Der Befehl

```
TOTAL ON <feld> TO <db-datei>
```

ermoeglicht das Verdichten von Datenbankdateien mit mindestens einem numerischen Feld. Voraussetzung fuer seine Anwendung ist, dass die aktivierte Datenbankdatei nach dem <feld> sortiert oder indiziert ist.

Es werden alle Saetze, die in dem spezifizierten Feld den gleichen Schluesselwert besitzen, zusammengefasst, die numerischen Felder ueber diese Saetze addiert und die zusammengefassten Saetze in eine neue Datenbankdatei, die durch "db-datei" spezifiziert wird, geschrieben.
Fuer die Struktur der Zieldatei wird die Struktur der aktivierten Ausgangsdatei uebernommen.
Die urspruengliche Datei bleibt nach Ausfuehrung des TOTAL-Befehles voll erhalten.

Im folgenden sollen jeweils alle Auftraege eines Kunden zusammengefasst werden. Dazu wird eine nach Kundennummern indizierte Auftragsdatei eroeffnet.

```
USE auftrag
INDEX ON kunr TO kunrauf
TOTAL ON kunr TO gesamt
```

REDABAS-4 erstellt die Datei "gesamt", in der alle Auftraege pro Kunde (d.h. mit gleicher Kundennummer) zusammengefasst sind. Nach Ausfuehrung des Befehls erscheint die Nachricht

```
11 Saetze addiert
 8 Saetze erzeugt
```

Das bedeutet, dass die urspruenglich 11 vorhandenen Datensaetze auf 8 komprimiert worden sind. Pro Kunde ist nur noch ein Datensatz vorhanden, dessen numerische Feldinhalte summiert wurden.

```
: USE gesamt
: LIST
 Satznr. AUFNR  KUNR     BETRAG    DATUM      BEZA
       1 2676   048         0.00     .  .      .F.
       2 2442   123       284.81   27.02.89    .F.
       3 0123   421        15.00   23.01.89    .T.
       4 0534   678       293.85   15.02.89    .T.
       5 6677   750      1420.00   17.02.89    .F.
       6 0211   789        38.18   23.02.89    .T.
       7 1234   850        35.54   27.02.89    .T.
       8 0310   910        73.02   02.02.89    .F.
```

Bild 40 Zusammenfassung der Auftragsdatei nach Kunden

Im Bild 40 ist die neue Datei "gesamt" aufgelistet. Sie zeigt - geordnet nach der Kundennummer - "betrag", "datum" und die Aussage, ob die Betraege bezahlt oder nicht bezahlt wurden, pro Kunde.
REDABAS-4 hat die Struktur der Datei "auftrag" automatisch uebernommen. Beachten Sie jedoch, dass beim Verdichten der Feldinhalt der nicht-numerischen Felder aus dem jeweils ersten Datensatz einer zusammengehoerigen Gruppe von Saetzen uebernommen wird. Dadurch liefern die Auftragsnummer, das logische Feld "beza" und auch das Datum keinerlei bzw. sogar verfaelschte Aussagen, da sich diese Informationen ja auf mehrere Auftraege beziehen.

Es ist also sinnvoll, solche Felder beim LIST-Befehl zu eliminieren.

Bild 41 demonstriert dies.

```
: LIST kunr,betrag
 Satznr.   kunr    betrag
       1    048       0.00
       2    123     284.81
       3    421      15.00
       4    678     293.85
       5    750    1420.00
       6    789      38.18
       7    850      35.54
       8    910      73.02
```

Bild 41 Anzeige der relevanten Felder der Datei "gesamt"

## 3.6. Die Arbeit mit Merkfeldern

Im Abschnitt 3.1.1. stellten wir die in REDABAS-4 moeglichen Datentypen vor. Felder vom Typ Zeichen (C), Numerisch (N), Logisch (L) und Datum (D) lernten Sie in den bisher erstellten und bearbeiteten Dateien bereits kennen.

Die tabellarische Darstellung der Daten - jede Tabellenzeile repraesentiert einen Datensatz aus fest formatierten Datenfeldern - kann aber mit REDABAS-4 auch wahlweise durch unformatierte Textinformationen ergaenzt werden.

So kann man in einer Datenbankdatei jedem Datensatz spezielle Textfelder zuordnen, die Zusammenfassungen, Kommentare, Erlaeuterungen aber auch die verbale Beschreibung von TGL-Blaettern, Richtlinien usw. enthalten koennen.
Dieser Datentyp heisst "MERK". Bei der Felddefinition druecken Sie bei Typ so lange die Leertaste, bis "MERK" angezeigt wird, oder Sie geben ein "M" ein. Die Feldlaenge von 10 Zeichen wird automatisch eingetragen, denn diese Laenge weisen alle Merkfelder in der Datenbankdatei auf, da dort nur ein Verweis gefuehrt wird. Der Text selbst wird in einer separaten Datei mit dem Datentyp .DBM gespeichert. Diese sogenannte Merkdatei wird von REDABAS-4 automatisch mit der zugehoerigen Datenbankdatei generiert, und beide Dateien werden stets gemeinsam eroeffnet.

Sie koennen eins oder auch alle Datenfelder (max. 128) einer Datenbankdatei als Merkfeld definieren. In einer Merkdatei (Dateityp .DBM) erfordern nur die Merkfelder Speicherplatz, in die tatsaechlich ein Text eingegeben wurde. Bei jeder Eingabe oder Aenderung von Text in ein Merkfeld werden in der Merkdatei jeweils 512-Bytes-Bloecke zugeordnet. In einem Merkfeld koennen maximal 4096 Bytes gespeichert werden, solange der REDABAS-4 - interne Texteditor nicht durch einen anderen Editor ausgetauscht wird (siehe dazu Parameter WP in Abschnitt 10.1.).

Die Dateneingabe in ein Merkfeld erfolgt mit dem REDABAS-4-Texteditor, der automatisch beim Aufruf eines Merkfeldes angesteuert wird. Dieser Texteditor arbeitet mit 66 eingebbaren Zeichen pro Zeile. Das letzte Wort bei einer Eingabe, das nicht mehr in die Zeile passt, wird in die naechste Zeile verlagert.

Geben Sie im Einfuegemodus in eine bereits bestehende Zeile Text ein, wird der bestehende Text ab Cursorposition nach rechts verschoben. Der dabei ueber eine Zeile gehende Text verschwindet. Solche Zeilen werden am rechten Rand durch ein Pluszeichen (+) gekennzeichnet. Sie koennen nach der Eingabe diese Zeile oder den gesamten Text mit <CTRL-K> und <B> neu formatieren (Reformat.).

Mit <CTRL-K> und <F> koennen Sie den auf dem Bildschirm anliegenden Text eines Merkfeldes nach Begriffen durchsuchen. Sie werden auf der 1. Bildschirmzeile aufgefordert, den Suchbegriff (ohne Hochkommas) einzugeben. Der Cursor wird an das 1. Zeichen des Suchbegriffes bewegt, wo der Suchbegriff im Text zum 1. Mal vorkommt. Mit <CTRL-K> und <L> wird auf der Bildschirmseite nach dem Suchbegriff weiter gesucht. Sie erhalten auf der 1. Bildschirmzeile eine Meldung, wenn der Suchbegriff nicht(mehr) vorkommt.

Sie koennen auch Daten einer externen Datei mit der Tastenkombination <CTRL-K> und <R> in das Merkfeld uebertragen. Das Betaetigen der Tastenkombination <CTRL-K> und <W> ermoeglicht, den Inhalt des Merkfeldes in eine externe Datei zu schreiben.

Beachten Sie bitte, dass die Merkfelder nur mit den bildschirmorientierten Eingabebefehlen APPEND, INSERT, CHANGE oder EDIT erstellt und editiert werden koennen. Den Inhalt eines Merkfeldes kann man nur mit den Befehlen ?, DISPLAY, LIST (der Feldname des Merkfeldes muss in den Befehlen explizit angegeben werden) oder REPORT FORM ausgeben.
Der Feldinhalt wird mit einer Zeilenlaenge von 50 Zeichen wiedergegeben. Diese Zeilenlaenge koennen Sie jedoch mit dem Befehl

```
SET MEMOWIDTH TO [<ausdrN>] veraendern, z.B.
SET MEMOWIDTH TO 66
```

(Die Zeilenlaenge stimmt mit der des Texteditors ueberein).

Wird mit COPY TO eine aktivierte Datenbankdatei kopiert, wird die zugehoerige Merkdatei automatisch mit uebertragen. Gleichzeitig wird beim Uebertragen die durch das Aendern von Merkfeldern bezueglich Speicherplatz vergroesserte Merkdatei auf volle Sektoren (nx512) verdichtet. Auch bei dem Befehl ZAP tritt eine Verdichtung auf, jedoch nicht bei den Befehlen PACK und SORT (vgl. Abschn. 9.3.3.).

Beachten Sie ausserdem, dass der Inhalt eines Merkfeldes nicht durchsucht werden kann, nicht einer Speichervariablen zugewiesen werden kann, keinen Ausdruck im Sinne von REDABAS-4 bilden kann, nicht fuer eine Bedingung innerhalb eines REDABAS-4-Befehls und fuer keine REDABAS-4-Funktion genutzt werden kann. Merkfelder lassen sich weder sortieren noch indizieren. Und denken Sie daran, dass eine Datenbankdatei mit Merkfeldern ohne die zugehoerige Merkdatei nicht genutzt werden kann, d.h. geht die Merkdatei verloren, ist die zugehoerige Datenbankdatei nicht mehr benutzbar.
Sie koennen die Daten Ihrer Datenbankdatei aber noch retten, indem Sie die Struktur unter einem anderen Namen und ohne Merkfelder neu definieren und mit dem Befehl APPEND FROM <datei> die Saetze der alten Datenbankdatei in die neue uebertragen.

Wenden Sie MODIFY STRUCTURE auf eine Datenbankdatei mit Merkfeldern an, und aendern Sie die Anzahl oder die Namen der Merkfelder, benennt REDABAS-4 den Namen der Merkdatei vom Dateityp .DBM auf den Dateityp .TBK um und benutzt den Inhalt zum Kopieren in die geaenderte Merkdatei vom Dateityp .DBM.
Dabei wird wie bei COPY TO die Merkdatei auf volle Sektoren verdichtet. Bei geringem Speicherplatz auf einem Datentraeger sollte beachtet werden, dass nach dem Modifizieren bzw. Kopieren Platz fuer eine 2. Datei zur Verfuegung stehen muss.

Die Arbeit mit Merkfeldern soll wieder an einem einfachen Beispiel demonstriert werden.
In die Kundendatei des Reparaturbetriebes sollen in ausgewaehlten Datensaetzen zusaetzliche Informationen in Form eines kurzen Textabschnitts aufgenommen werden.
Die Datei soll "kunden1" heissen. Kopieren Sie dazu die vorhandene Datei "kunden":

```
USE kunden
COPY TO kunden1
```

Geben Sie die Befehle

```
USE kunden1
MODIFY STRUCTURE
```

ein und bewegen Sie den Cursor in der bereitgestellten Maske mit der Pfeiltaste unter das letzte Datenfeld (strasse). Spezifizieren Sie ein zusaetzliches Datenfeld vom Typ "MERK" mit dem Namen "info". Sie erkennen, dass bei "Laenge" von REDABAS-4 automatisch "10" eingetragen wird.
Ueberzeugen Sie sich mit

```
LIST STRUCTURE
```

von der Richtigkeit Ihrer Eingaben und lassen Sie sich die Struktur der Datei "kunden1" anzeigen, die Bild 42 zeigt.

```
: LIST STRUCTURE
Datenbankdateistruktur :  A:kunden1.dbd
Anzahl der Datensaetze :        13
Letztes Aenderungsdatum:  09.11.89
Feld     Feldname      Typ          Laenge          Dez
    1    KUNR          Zeichen           3
    2    NAME          Zeichen          10
    3    VORNAME       Zeichen           8
    4    PLZ           Zeichen           4
    5    ORT           Zeichen           8
    6    STRASSE       Zeichen          15
    7    INFO          MERK             10
 * Gesamt *                            59
```

Bild 42  Struktur der Datenbankdatei "kunden1" (mit Merkfeld)

In das Merkfeld namens "info" sollen nun Zusatzinformationen fuer bestimmte Kunden eingetragen werden. Wir nutzen dafuer den bekannten EDIT-Befehl.

Rufen Sie mit

```
EDIT 1
```

den ersten Datensatz zur Bearbeitung auf. Bewegen Sie den Cursor in das Datenfeld "info". Druecken Sie die Tastenkombination <CTRL-Home> und Sie sehen, dass der Texteditor aufgerufen wird.
Fuer die Eingabe des Textes koennen Sie wieder alle Editierfunktionen nutzen. Die Texteingabe wird mit <CTRL-End> abgeschlossen, und der Text wird gespeichert. REDABAS-4 kehrt nun zur EDIT-Maske zurueck, und Sie koennen nacheinander die naechsten Datensaetze zur Eingabe von Text in das Feld "info" aufrufen.

Fuer die Ausgabe des Textes nutzen wir den LIST-Befehl, der selektiv angewendet werden muss.
Wir waehlen fuer die Ausgabe Kundennummer, Name und die Zusatzinformationen und moechten auf die Ausgabe der Satznummer verzichten. Die entsprechende Formulierung des Befehls und das Ergebnis zeigt Bild 43.
Wir moechten Sie darauf hinweisen, dass wir fuer unser Beispiel zur Demonstration nur relativ kurze Informationen waehlten, fuer die sich das Anlegen eines Merkfeldes eigentlich nicht lohnt.

```
: LIST kunr,name,info OFF
 kunr name        info
 123  Schmidt     Diese Kundin ist sehr stark behindert
                  (Rollstuhlfahrerin).
                  Reparaturen sind nach Anmeldung unver-
                  zueglich durchzufuehren.

 366  Schmidt     Das Dienstfahrzeug (Wartburg Tourist
                  YE 12-21) wurde mit Anhaengerkupplung
                  versehen.

 850  Weber
 910  Lolle       Bei Reparaturen ist mit der vorherigen
                  Reparaturwerkstatt des Kunden (Autoservice
                  Meissen) unbedingt Ruecksprache zu nehmen.

 421  Meier       Letzte Reparatur am 17.1.89. Getriebe-
                  wechsel wurde bereits vorgemerkt.

 789  Neumann     Trabant REH 76-89 entspricht nicht mehr
                  Sicherheitsvorschriften. Kundin wurde
                  bereits mehrfach darauf hingewiesen.

 750  Fleischer
 678  Miller      Letzte Durchsicht des Trabant YE 87-99
                  am 1.2.89.

 645  Moser       Glysanthin abgefuellt am 1.1.89.

 083  Heinze      Fahrzeugpapiere des Kunden entsprechen
                  nicht den geltenden Vorschriften. Der
                  Kunde wurde bereits mehrfach darauf hin-
                  gewiesen.
 048  Lehmann     Letzte Generalreparatur am 30.1.89.
 267  Schulze     Einbau einer Anhaengerkupplung am 1.2.89.
                  Letzte grosse Durchsicht am 15.2.89.

 999  Tauber
```

Bild 43  Anzeige von Merkfeldern

Wollen Sie die gesamte Datei "kundenl" auflisten, sehen Sie, dass die Merkfelder selbst nicht angezeigt werden, sondern -->M als Verweis auf die dazugehoerige Merkdatei.
Bild 44 zeigt Ihnen dies.

```
: USE kundenl
: LIST OFF
```

| KUNR | NAME | VORNAME | PLZ | ORT | STRASSE | INFO |
|---|---|---|---|---|---|---|
| 123 | Schmidt | Gisela | 8223 | THARANDT | Bahnhofstr. 43 | -->M |
| 366 | Schmidt | Heinz | 8223 | THARANDT | Waldweg 42 | -->M |
| 850 | Weber | Franz | 8223 | THARANDT | Heideweg 66 | -->M |
| 910 | Lolle | Peter | 8122 | RADEBEUL | Goethestr. 13 | -->M |
| 421 | Meier | August | 8122 | RADEBEUL | Uferstr. 12 | -->M |
| 789 | Neumann | Inge | 8222 | RABENAU | Gartenstr. 4 | -->M |
| 750 | Fleischer | Karl | 8212 | FREITAL | Am See 83 | -->M |
| 678 | Miller | Peter | 8212 | FREITAL | Nelkenweg 23 | -->M |
| 645 | Moser | Otto | 8212 | FREITAL | Neugraben 8 | -->M |
| 083 | Heinze | Max | 8060 | DRESDEN | Rosenstr. 78 | -->M |
| 048 | Lehmann | Kurt | 8060 | DRESDEN | Landstr. 15 | -->M |
| 267 | Schulze | Ernst | 8060 | DRESDEN | Nebenstr. 61 | -->M |
| 999 | Tauber | Fred | 8028 | DRESDEN | Fliederstr. 42 | -->M |

Bild 44 Datenbankdatei "kundenl"

## 3.7. Reportgenerator

REDABAS-4 bietet mit dem Reportgenerator ein Unterprogramm, das die Aufbereitung und Ausgabe von Listen (Reports), die Nutzerwuenschen entsprechend gestaltet werden koennen, auf dem Drucker und/oder Bildschirm erlaubt, wobei das Reportformat fuer wiederholten Aufruf gespeichert wird. Von REDABAS-4 wird eine Reihe von Masken fuer das Festlegen des Reportformats bereitgestellt, der Nutzer gibt lediglich die entsprechenden Parameter ein. Damit wird das aufwendige Programmieren von Drucklisten eingespart. Mit dem Reportgenerator haben Sie also die Moeglichkeit, zu bestimmen, welche der Daten Ihrer Datei auf welche Weise aufgelistet werden sollen. Ausserdem koennen Sie Summen und Gruppensummen (Zwischensummen je Gruppe) bilden, Ueberschriften erstellen, die Druckseiten formatieren und mit Seitennummer und aktuellem Datum versehen. Haben Sie das Listenformular Ihren Wuenschen entsprechend aufbereitet, wird es als sogenannte Report-Datei (Dateityp .DEF) gespeichert und ist jederzeit abrufbar. Nachtraegliche Veraenderungen an einem Listenformular sind problemlos moeglich.

Bevor wir Ihnen anhand von Beispielen die Arbeit mit dem Reportgenerator demonstrieren, moechten wir zusaetzlich zu den bereits vorhandenen Datenbankdateien "kunden" und "auftrag" eine weitere Datei, genannt "bestand", einfuehren. Diese Datei soll den Lagerbestand beinhalten und enthaelt Informationen ueber die Teilnummer (teilnr), die Bezeichnung eines Teils (bezei), die Anzahl (anz) und den Preis (preis) sowie Lagerort (ortnr), Datum (datum) und den Mindestbestand (minbest).

Bild 45 zeigt die Struktur dieser Datenbankdatei und Bild 46 die Datei "bestand" selbst.

Datenbankdateistruktur : A:bestand.dbd
Anzahl der Datensaetze : 20
Letztes Aenderungsdatum: 09.09.89

| Feld | Feldname | Typ | Laenge | Dez |
|---|---|---|---|---|
| 1 | TEILNR | Zeichen | 5 | |
| 2 | BEZEI | Zeichen | 17 | |
| 3 | ANZ | Numerisch | 4 | |
| 4 | PREIS | Numerisch | 7 | 2 |
| 5 | ORTNR | Zeichen | 3 | |
| 6 | DATUM | Datum | 8 | |
| 7 | MINBEST | Numerisch | 4 | |
| * Gesamt * | | | 49 | |

Bild 45 Struktur der Datenbankdatei "bestand"

| Satznr. | TEILNR | BEZEI | ANZ | PREIS | ORTNR | DATUM | MINBEST |
|---|---|---|---|---|---|---|---|
| 1 | 00031 | Sechskantmutter | 5972 | 0.56 | 110 | 18.01.89 | 2400 |
| 2 | 00065 | Zylinderschraube | 3472 | 0.21 | 110 | 18.01.89 | 1400 |
| 3 | 00106 | Federring | 1374 | 0.03 | 110 | 18.01.89 | 2000 |
| 4 | 00108 | Dichtring | 766 | 1.90 | 110 | 18.01.89 | 320 |
| 5 | 00327 | Keilriemen | 233 | 6.10 | 120 | 18.01.89 | 120 |
| 6 | 00353 | Scheinwerfer | 30 | 31.60 | 130 | 11.02.89 | 15 |
| 7 | 00425 | Bremstrommel | 10 | 14.60 | 140 | 18.01.89 | 120 |
| 8 | 00516 | Batterie | 3 | 182.00 | 150 | 11.02.89 | 5 |
| 9 | 00730 | Lenkstock,vollst. | 60 | 44.00 | 130 | 11.02.89 | 25 |
| 10 | 00789 | Seilzug | 295 | 3.10 | 120 | 18.01.89 | 120 |
| 11 | 00819 | Bremsleitung | 31 | 5.80 | 140 | 18.01.89 | 150 |
| 12 | 00852 | Bolzen | 471 | 0.10 | 110 | 18.01.89 | 600 |
| 13 | 14043 | Ansaugkruemmer | 194 | 2.45 | 140 | 18.01.89 | 80 |
| 14 | 17000 | Blinkleuchte,vorn | 114 | 10.00 | 130 | 18.01.89 | 50 |
| 15 | 30000 | Bremszylinder | 278 | 9.35 | 140 | 18.01.89 | 110 |
| 16 | 30027 | Radzylinder | 6 | 10.50 | 130 | 18.01.89 | 20 |
| 17 | 31100 | Kolben | 200 | 27.80 | 120 | 11.02.89 | 80 |
| 18 | 31200 | Keilriemenscheibe | 253 | 10.90 | 120 | 18.01.89 | 100 |
| 19 | 40018 | Motor | 7 | 1420.00 | 150 | 18.01.89 | 30 |
| 20 | 40359 | Kurbelwelle | 63 | 280.00 | 140 | 18.01.89 | 30 |

Bild 46 Datenbankdatei "bestand"

Der Befehl zum Aufruf des Reportgenerators lautet in vereinfachter Form

```
CREATE REPORT <reportdatei>
```

<reportdatei> ist ein frei waehlbarer, jedoch auf den Inhalt Bezug nehmender Name, unter dem die Listenstruktur als Datei gespeichert wird und auf Wunsch wieder aufgerufen werden kann. REDABAS-4 fuegt intern die Typbezeichnung .DEF an den Dateinamen an.

Bevor der Reportgenerator aufgerufen wird, muss die auszuwertende Datei vorher eroeffnet werden.

Es soll jetzt eine Liste des Lagerbestandes erstellt werden, wobei ausser Teilnummer, Bezeichnung und Preis der aktuelle Lagerbestand und der konstante Wert des Mindestlagerbestandes enthalten sein sollen.
Mit den Befehlen

```
USE bestand
CREATE REPORT best1
```

wird die Datei "bestand" aufgerufen und der Name "best1" fuer eine Reportdatei vergeben.

REDABAS-4 stellt Ihnen jetzt Menues bereit, in denen Sie je nach Aufforderung Eintragungen bzw. eine Auswahl vornehmen koennen. Benutzen Sie dabei die vom Editieren bekannten Tastenkombinationen (vgl. Abschnitt 3.1.3.) und die im Seitenmodus gueltigen Tastenkombinationen (vgl. Abschn. 8.4.2.). Insbesondere sind die Unten- bzw. Oben-Pfeiltasten (oder <PgDn> bzw. <PgUp> zur Grobeinstellung) zur Fuehrung des Markierungsbalkens auf eine Option und anschliessend <ET> zur Auswahl dieser Option zu betaetigen. Mit den Rechts- und Links-Pfeiltasten erreichen Sie die 5 Menues des Reportgenerators.

Im unteren Teil einiger Menues wird alternativ das bisher definierte Reportformat oder die Hilfstafel mit den Tastenbelegungen zur Cursorbewegung - jeweils durch Betaetigen der <F1>-Taste gesteuert - angezeigt.

Bild 47 zeigt das Format-Menue des Reportgenerators mit der Anzeige der Hilfstafel.

```
-------------------------------------------------------------
|Format    Gruppe    Spalte    Auswahl   Ende        19.30.05|
|-------------------------------------                        |
|| Ueberschrift                    >    |                     |
|| Seitenbreite(Stellen)            80  |---------------------|
|| Linker Rand(Stellen)              8  ||Lagerbestand       ||
|| Rechter Rand(Stellen)             0  ||                   ||
|| Zeilen je Seite                  58  ||                   ||
|| Doppelzeiliger Report           Nein ||                   ||
|| Seitenvorschub vor  dem Druck Ja    |---------------------|
|| Seitenvorschub nach dem Druck Nein  |                     |
|| Kopf einfach                    Nein |                     |
|-------------------------------------                        |
|                                                             |
|-------------------------------------------------------------|
||Cursor:<-- -->|Loe. Zeich.:Del|Einf. Spal.:^N|Einf.:   Ins||
||Zeich.: <- -> |Loe. Wort  :^T |Hilfe/Form.:F1|Zoom e:^PgDn||
||Wort: Home End|Loe. Spalte:^U |Abbruch:   ESC|Zoom a:^PgUp||
|-------------------------------------------------------------|
|                                                             |
|                                                             |
|CREATE REPORT|<A:>|BEST1.DEF   |Opt: 1/9       |      |      |
|  Eingabe der Report-Ueberschrift. Ende - CTRL-End.          |
|Ueberschriftszeilen, die auf jeder Seite auszugeben sind.    |
-------------------------------------------------------------
```

Bild 47 Format-Menue des Reportgenerators

Durch Auswahl der 1. Option wird die (gegebenenfalls mehrzeilige) Ueberschrift der Liste eingetragen. Ausserdem enthaelt das Menue die Formatparameter der Liste. Sie koennen die angebotenen Werte durch Uebergehen mit der Oben- bzw. Unten-Pfeiltaste uebernehmen oder wunschgemaess durch Auswahl mit der <ET>-Taste und gegebenenfalls nachfolgender Eingabe ueberschreiben. Die Ueberschrift wird automatisch zentriert.

Tragen Sie in das rechte hell unterlegte Feld (vgl. Bild 47) die Ueberschrift ein:

Lagerbestand

und geben Sie die Parameter so ein, wie sie im Bild 48 eingetragen sind. Beachten Sie das Abschliessen der Werte mit den in der Anweisungszeile ausgewiesenen Tasten.

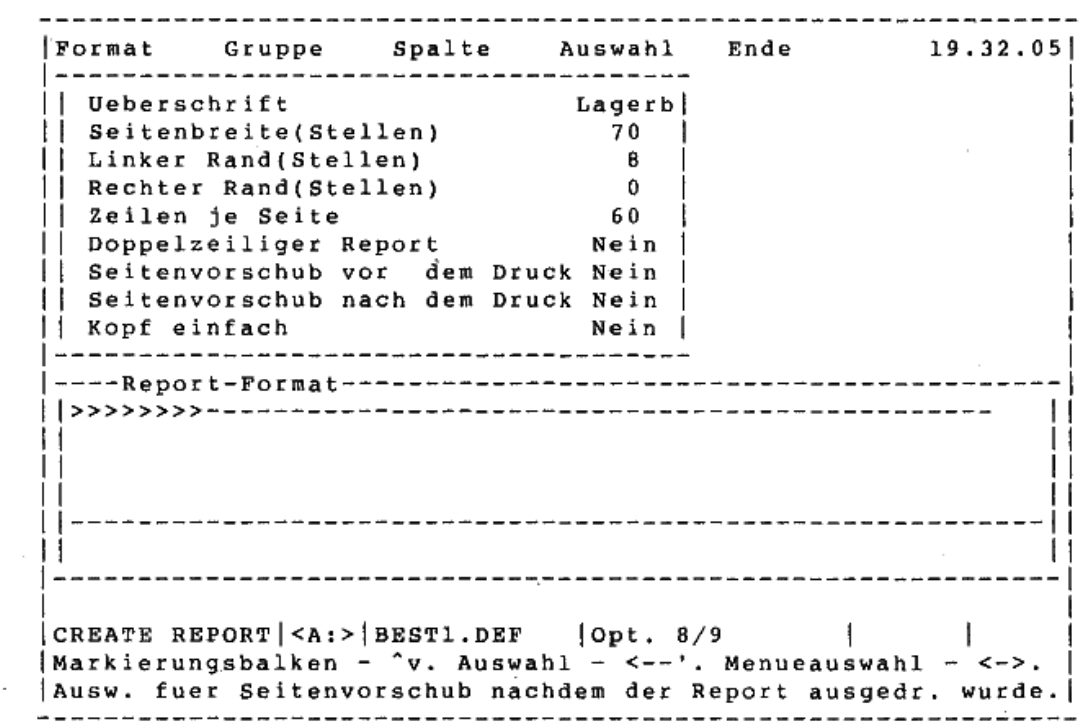



Bild 48 Format-Menue des Reportgenerators fuer Liste "Lagerbestand"

Die Hilfstafel wurde mit <F1> bereits durch das Report-Format ersetzt. Haben Sie die letzte zu aendernde Option innerhalb dieses Menues durch <ET> verneint, gehen Sie mit der Rechts-Pfeiltaste zum Gruppe-Menue. In diesen Menue wird festgelegt, ob bzw. fuer welche Felder Gruppensummen gebildet werden sollen.

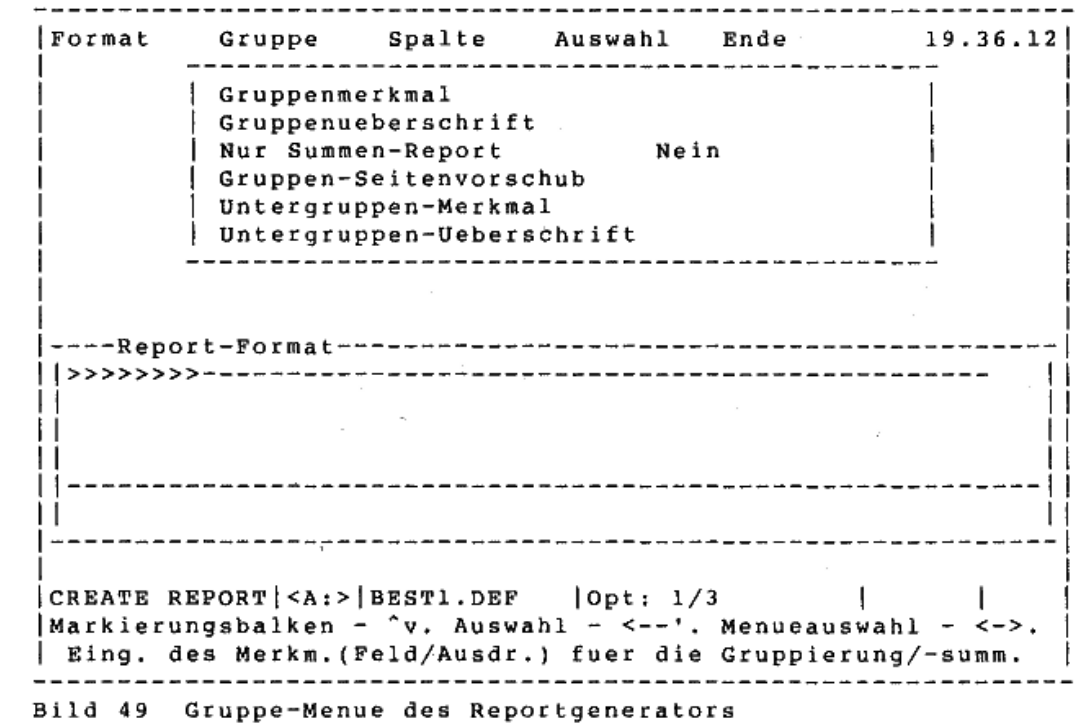



Bild 49 Gruppe-Menue des Reportgenerators

Im ersten Beispiel der Arbeit mit dem Reportgenerator sollen noch keine Gruppensummen gebildet werden.
Wir brauchen das Menue also nicht zu durchlaufen und springen durch Betaetigen der Rechts-Pfeiltaste sofort in das Spalte-Menue, das in Bild 50 dargestellt ist.

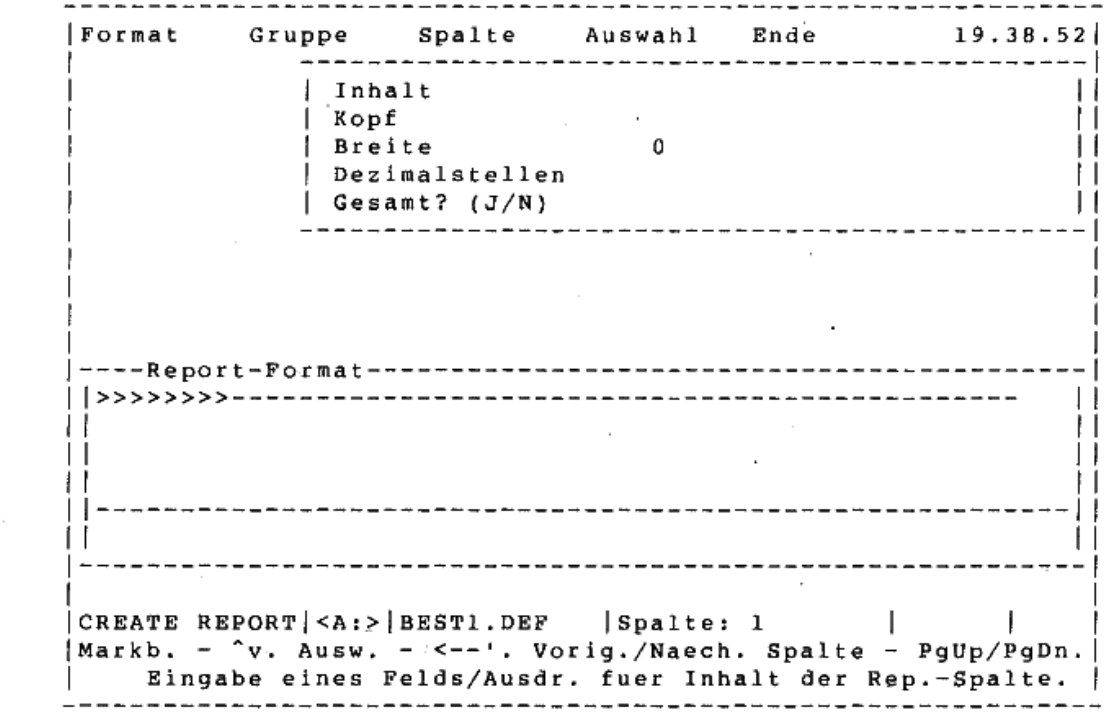


Bild 50 Spalte-Menue des Reportgenerators

Dieses Menue dient der eigentlichen Beschreibung der Liste, die ja in Form einer Tabelle angelegt ist. Sie muessen jetzt praktisch pro Tabellenspalte den Inhalt und die Ueberschrift (Kopf) festlegen. Fuer diese beiden Spezifikationen wird pro Spalte ein neues Spalte-Menue angeboten.
Gleichzeitig legen Sie - je nachdem mit welchem Datenfeld Sie beginnen und welches Datenfeld Sie aufnehmen - die Reihenfolge und Anzahl der Spalten innerhalb Ihrer Tabelle fest.

In Bild 50 erkennen Sie in der Statuszeile, dass das erste Spalte-Menue mit Spalte 1 bezeichnet wird.

Waehlen Sie die Option Inhalt, tragen Sie in das hell markierte Feld den Feldnamen "teilnr" ein, und schliessen Sie dies mit <ET> ab!
Sie sehen, dass REDABAS-4 bei Breite automatisch eine 5 eingetragen hat - entsprechend der definierten Datenfeldlaenge fuer das Feld "teilnr" (vgl. Bild 45).

Mit der Unten-Pfeiltaste und anschliessendem <ET> gelangen Sie in die naechste Option namens "Kopf". Es wird ein vierzeiliges Eingabefenster bereitgestellt. Sie schreiben in die 1. Zeile

Teile-Nr

und druecken <CTRL-End>.

Bei Breite wurde jetzt automatisch auf 8 erhoeht, naemlich auf die Buchstabenanzahl des Wortes Teile-Nr. Sie koennen dort auch selbst die gewuenschte Spaltenbreite eintragen. REDABAS-4 setzt automatisch ein Leerzeichen zwischen die Spalten. Die Eintragung Gesamt? (J/N) in der letzten Zeile der Maske bezieht sich auf die Bildung von Gesamtsummen. Bild 51 zeigt die ersten Eintragungen.

In jedem neuen Spalten-Menue wird ein Abbild des aktuellen Listenaufbaus angezeigt. Es werden die bis dahin vergebenen Spaltenbezeichnungen, der verbleibende Platz in der Zeile (durch eine Strichlinie angedeutet) und darunter in symbolischer Darstellung der Spalteninhalt angezeigt.

In der symbolischen Darstellung bedeuten

```
     XXX... Feld vom Typ Zeichen
     ###... numerisches  Feld mit Gesamtsummenbildung
     999... numerisches Feld.
```

```
-------------------------------------------------------------
|Format    Gruppe    Spalte    Auswahl   Ende        19.39.10|
|              ----------------------------------------------|
|              | Inhalt           teilnr                     ||
|              | Kopf             Teile-Nr                   ||
|              | Breite             8                        ||
|              | Dezimalstellen                              ||
|              | Gesamt? (J/N)                               ||
|              ----------------------------------------------|
|                                                            |
|                                                            |
|                                                            |
|                                                            |
|----Report-Format-----------------------------------------|
||>>>>>>>>Teile-Nr ---------------------------------------  ||
||                                                          ||
||                                                          ||
||                                                          ||
||-----------------------------------------------------------||
||        XXXXX                                             ||
|-----------------------------------------------------------|
|                                                            |
|CREATE REPORT|<A:>|BEST1.DEF   |Spalte: 1       |      |    |
|Markb. - ^v. Ausw. - <--'. Vorig./Naech. Spalte - PgUp/PgDn.|
|    Eingabe der Ausgabe-Breite dieser Report-Spalte.        |
-------------------------------------------------------------
```

Bild 51 Definition der ersten Spalte der Liste "Lagerbestand"

Damit wurde die erste Spalte der Liste definiert. Druecken Sie die <PgDn>-Taste, dann erhalten Sie das Spalte-Menue fuer die naechste Spaltendefinition.
Pro Spalte-Menue sind fuer die einzelnen Spalten unserer Liste folgende Eintragungen erforderlich:

```
Spalte Inhalt                        Kopf                 Gesamt
-------------------------------------------------------------------
  1    teilnr (schon eingetragen)  Teile-Nr
  2    bezei                       Bezeichnung
  3    anz                         akt. Stand
  4    minbest                     min. Stand
  5    preis                       Preis                  N
```

Bei den drei numerisch definierten Feldern wird nach der Definition des Inhalts und Kopfes automatisch die Frage bejaht, ob eine Gesamtsumme gebildet werden soll.

Bild 52 zeigt das Spalte-Menue nach der letzten Eintragung "Gesamt? (J/N)" zu "preis" in unserer Tabelle. Nachdem hier die letzte Eintragung mit <ET> verneint wurde, kommen Sie durch zweimaliges <->> in das Ende-Menue. Waehlen Sie nun die Option Speichern mit <ET>!

Ist die Definition abgeschlossen, erfolgt die Generierung und Speicherung des Listenformats als Reportdatei.

```
-------------------------------------------------------------------
|Format    Gruppe    Spalte    Auswahl   Ende        19.43.16|
|              ----------------------------------------------|
|              | Inhalt            preis                      ||
|              | Kopf              Preis                      ||
|              | Breite              7                        ||
|              | Dezimalstellen      2                        ||
|              | Gesamt? (J/N)     Nein                       ||
|              ----------------------------------------------|
|                                                             |
|                                                             |
|                                                             |
|                                                             |
|----Report-Format-------------------------------------------|
||e-Nr Bezeichnung       akt. Stand min. Stand Preis   --    ||
||                                                           ||
||                                                           ||
||                                                           ||
||-----------------------------------------------------------||
||X    XXXXXXXXXXXXXXXXX       ####       #### 9999.99       ||
|-------------------------------------------------------------|
|                                                             |
|CREATE REPORT|<A:>|BEST1.DEF   |Spalte: 5       |      |     |
|Markierungsbalken - ^v. Auswahl - <--'. Menueauswahl - <->-  |
|  Soll fuer dieses Feld eine Gesamtsumme ausgegeben werden ?|
-------------------------------------------------------------------
```

Bild 52 Eintragung der Spalte "Preis"

Seite 1
09.09.89

Lagerbestand

| Teile-Nr | Bezeichung | akt.Stand | min. Stand | Preis |
|---|---|---|---|---|
| 00031 | Sechskantmutter | 5972 | 2400 | 0.56 |
| 00065 | Zylinderschraube | 3472 | 1400 | 0.21 |
| 00106 | Federring | 1374 | 2000 | 0.03 |
| 00108 | Dichtring | 766 | 320 | 1.90 |
| 00327 | Keilriemen | 233 | 120 | 6.10 |
| 00353 | Scheinwerfer | 30 | 15 | 31.60 |
| 00425 | Bremstrommel | 10 | 120 | 14.60 |
| 00516 | Batterie | 3 | 5 | 182.00 |
| 00730 | Lenkstock,vollst. | 60 | 25 | 44.00 |
| 00789 | Seilzug | 295 | 120 | 3.10 |
| 00819 | Bremsleitung | 31 | 150 | 5.80 |
| 00852 | Bolzen | 471 | 600 | 0.10 |
| 14043 | Ansaugkruemmer | 194 | 80 | 2.45 |
| 17000 | Blinkleuchte,vorn | 114 | 50 | 10.00 |
| 30000 | Bremszylinder | 278 | 110 | 9.35 |
| 30027 | Radzylinder | 6 | 20 | 10.50 |
| 31100 | Kolben | 200 | 80 | 27.80 |
| 31200 | Keilriemenscheibe | 253 | 100 | 10.90 |
| 40018 | Motor | 79 | 30 | 1420.00 |
| 40359 | Kurbelwelle | 63 | 30 | 280.00 |
| *** Gesamt *** | | 13904 | 7775 | |

Bild 53 Mit dem Reportgenerator erstellter Lagerbestand (Reportdatei "best1")

Die Liste ist jetzt unter dem Namen "best1" als Reportdatei jederzeit abrufbar, beispielsweise nach dem Aktualisieren der Reportdatei oder der Datenbank. Der Definitionsschritt entfaellt dann, es sind nur die beiden Befehle zum Eroeffnen der Datei und zum Aufruf des Reportgenerators erforderlich:

```
USE bestand
REPORT FORM best1 TO PRINT
```

Der Zusatz TO PRINT bewirkt die Ausgabe der Liste auf dem Drucker, unabhaengig davon, ob der Drucker schon mit <CTRL-P> eingeschaltet war.

Bild 53 zeigt die Liste "Lagerbestand". Wie Sie sehen, erscheinen Seitennummer und das vor dem Starten von REDABAS-4 eingegebene aktuelle Datum automatisch.

Allgemein ist zu sagen, dass innerhalb des Listenformulars nur der Name von Datenfeldern, aber nicht der Dateiname selbst erscheint. Das resultiert daraus, dass die Erstellung der Liste nicht an eine bestimmte Datei gebunden ist. Grundlage fuer den Listenaufbau koennen alle diejenigen Dateien sein, die die benoetigten Felder enthalten. (Die Felder muessen bezueglich Name, Typ und Laenge uebereinstimmen.) Ausserdem werden die verschiedenen, zu einer Stammdatei gehoerenden Indexfolgen oft Grundlage fuer die Erstellung ein- und desselben Listenformats sein.

Mit dem Befehl

MODIFY REPORT <reportdatei>

kann die Reportdatei jederzeit geaendert werden. Nach Eingabe des Befehls MODIFY REPORT werden die einzelnen Masken in ihrer aktuellen Belegung wieder bereitgestellt, und Sie koennen Eintragungen ueberschreiben, loeschen oder hinzufuegen.
Verwenden Sie

| | |
|---|---|
| <PgDn> | fuer das naechste Spalte-Menue |
| <PgUp> | fuer das vorhergehende Spalte-Menue |
| <CTRL-U> | fuer das Loeschen einer Spalte |
| <CTRL-N> | fuer das Einfuegen einer Spalte |

Auf der naechsten Liste soll der wertmaessige Lagerbestand, zusammengefasst je Lagerort, ausgegeben werden. Neben den schon bekannten Spalten kommt die Spalte Gesamtwert hinzu, wobei die Feldinhalte von Preis und Anzahl (hier mit "Einzelpreis" und "Bestand" bezeichnet) miteinander multipliziert werden.

Rufen Sie den Reportgenerator auf mit

CREATE REPORT best21

Sie bekommen - genau wie im vorigen Beispiel - nacheinander die Menues fuer das Eintragen von Ueberschrift und Formatparametern, das Bilden von Gruppen- und Untergruppensummen, die einzelnen Spaltenbeschreibungen und das Ende durch die entsprechenden Tastenbetaetigung eingeblendet.
Im Format-Menue geben Sie als Ueberschrift ein:

Wertmaessiger Lagerbestand
(nach Lagerorten aufgeschluesselt)

Die Ueberschrift kann maximal vier Zeilen einnehmen, eine mittige Plazierung erfolgt automatisch. Die Formatparameter uebernehmen Sie von REDABAS-4.

Das Gruppe-Menue (s. Bild 49) muessen wir jetzt ausfuellen, denn es sollen fuer jeden Lagerort Gruppensummen ausgegeben werden.
Schreiben Sie unter

Gruppenmerkmal: ortnr
(Nur Summen-Report und
Gruppen-Seitenvorschub sind auf Nein eigestellt.)
Gruppenueberschrift: Lagerort

Die anderen Angaben werden nicht ausgefuellt, mit <->> kommen Sie zum Spalte-Menue fuer die erste Spaltendefinition und anschliessend, mit <PgDn> zu den weiteren. Fuellen Sie bitte (fuer jedes Feld ein Spalte-Menue) aus:

| Spalte | Inhalt | Kopf | Ges |
|---|---|---|---|
| 1 | teilnr | Teile-<br>Nr | |
| 2 | bezei | Bezeichnung | |
| 3 | preis | Einzel-<br>preis | N |
| 4 | anz | Bestand | N |
| 5 | preis*anz | Gesamt-<br>wert | |

Im letzten Feld wurde anstatt eines Feldnamens eine Formel eingetragen. Dies ist ohne weiteres moeglich.
Uebrigens wurde das hell unterlegte Feld fuer Eintragungen deshalb so gross gewaehlt, weil dort auch Ausdruecke groesseren Ausmasses Platz finden koennen.
Schliessen Sie die Listendefinition mit der Option Speichern des Ende-Menues ab!
Da eine Summenbildung je Lagerort bei der Listenausgabe gewuenscht ist, muss die Datei nach diesem Datenfeld geordnet (sortiert oder indiziert) vorliegen. Der Aufbau einer entsprechenden Indexdatei ist zweckmaessig. Auf bekannte Weise wird deshalb die Indexdatei "ortbest" erstellt und als Ausgangsdatei verwendet.

```
USE bestand
INDEX ON ortnr TO ortbest
REPORT FORM best21 TO PRINT
```

Bild 54 zeigt die entstandene Liste.

Wertmaessiger Lagerbestand
(nach Lagerorten aufgeschluesselt)

| Teile-Nr | Bezeichnung | Einzelpreis | Bestand | Gesamtwert |
|---|---|---|---|---|
| ** Lagerort 110 | | | | |
| 00031 | Sechskantmutter | 0.56 | 5972 | 3344.32 |
| 00065 | Zylinderschraube | 0.21 | 3472 | 729.12 |
| 00106 | Federring | 0.03 | 1374 | 41.22 |
| 00108 | Dichtring | 1.90 | 766 | 1455.40 |
| 00852 | Bolzen | 0.10 | 471 | 47.10 |
| ** Gruppensumme ** | | | | 5617.16 |
| ** Lagerort 120 | | | | |
| 00327 | Keilriemen | 6.10 | 233 | 1421.30 |
| 00789 | Seilzug | 3.10 | 295 | 914.50 |
| 31100 | Kolben | 27.80 | 200 | 5560.00 |
| 31200 | Keilriemenscheibe | 10.90 | 253 | 2757.70 |
| ** Gruppensumme ** | | | | 10653.50 |
| ** Lagerort 130 | | | | |
| 00353 | Scheinwerfer | 31.60 | 30 | 948.00 |
| 00730 | Lenkstock,vollst. | 44.00 | 60 | 2640.00 |
| 17000 | Blinkleuchte, vorn | 10.00 | 114 | 1140.00 |
| 30027 | Radzylinder | 10.50 | 6 | 63.00 |
| ** Gruppensumme ** | | | | 4791.00 |
| ** Lagerort 140 | | | | |
| 00425 | Bremstrommel | 14.60 | 10 | 146.00 |
| 00819 | Bremsleitung | 5.80 | 31 | 179.80 |
| 14043 | Ansaugkruemmer | 2.45 | 194 | 475.30 |
| 30000 | Bremszylinder | 9.35 | 278 | 2599.30 |
| 40359 | Kurbelwelle | 280.00 | 63 | 17640.00 |
| ** Gruppensumme ** | | | | 21040.40 |
| ** Lagerort 150 | | | | |
| 00516 | Batterie | 182.00 | 3 | 546.00 |
| 40018 | Motor | 1420.00 | 79 | 112180.00 |
| ** Gruppensumme ** | | | | 112726.00 |
| *** Gesamt *** | | | | 154828.06 |

Bild 54 Liste "Wertmaessiger Lagerbestand", nach Lagerorten aufgeschluesselt, mit Einzelpositionen (Reportdatei "best21")

Will man sehr grosse Dateien auswerten, ist es zweckmaessig, auf den Druck der einzelnen Positionen zu verzichten und nur die Gruppensummen und die Summe einer Liste auszugeben. In diesem Fall muss die Frage:

Nur Summen-Report

mit <ET> bejaht werden (vgl. Bild 49).

Bild 55 zeigt eine solche verkuerzte Form einer Liste (Reportdatei "best22").
Wir moechten den Aufbau dieser Liste aus dem vorigen Listenaufbau entwickeln. Dazu kopieren wir die Reportdatei best21 in eine Reportdatei best22 und nehmen in dieser Datei die entsprechenden Veraenderungen vor. Schreiben Sie folgenden Befehl:

COPY FILE best21.def TO best22.def

Sie lernten den COPY-Befehl schon im Abschnitt 3.4.1. kennen. Fuer das Kopieren einer geoeffneten Datenbankdatei (Typ .DBD) genuegte das Befehlswort COPY. Will man Dateien beliebiger Typen kopieren, muss der Befehl auf COPY FILE erweitert werden. Ausserdem muessen Quell- und Zieldatei mit Dateityp angegeben werden.

Als Ergebnis dieses Kopierbefehls liegt die Reportdatei best22 mit gleichem Aufbau wie best21 vor. Mit

MODIFY REPORT best22

rufen wir den Reportgenerator auf, um Aenderungen vornehmen zu koennen. Auf dem Bildschirm erscheint das Format-Menue.
Veraendern Sie lediglich

Seitenbreite: 55
Linker Rand: 10

Blaettern Sie mit <->> zum Gruppe-Menue.

Veraendern Sie

Nur Summen-Report auf Ja durch Druecken von <ET>

Springen Sie weiter zum Spalte-Menue und loeschen Sie mit <CTRL-U> die Eintragungen fuer Spalte 1.

Fuehren Sie Weiterblaettern mit <PgDn> und Loeschen fort, bis Sie bei Spalte 5 angekommen sind (d. h. Loeschen der ersten vier Spalten).
Im Spalte-Menue fuer Spalte 5 (preis*anz) uebernehmen Sie alle Eintragungen und ueberschreiben lediglich
Breite: 40.

Damit ist das neue Listenformat erstellt. Sie koennen den Reportgenerator verlassen, indem Sie die Option Speichern im Ende-Menue auswaehlen.

Ausgangspunkt fuer den Aufruf der Liste ist wieder die Indexdatei "ortbest". Mit der Befehlsfolge

```
USE bestand
INDEX ON ORTNR TO ortbest
REPORT FORM best22 TO PRINT
```

wird die Liste erzeugt, die Bild 55 zeigt.

```
Seite  1
09.09.89
          Wertmaessiger Lagerbestand
      (nach Lagerorten aufgeschluesselt)

                                              Gesamtwert

** Lagerort 110
** Gruppensumme **                              5617.16

** Lagerort 120
** Gruppensumme **                             10653.50

** Lagerort 130
** Gruppensumme **                              4791.00

** Lagerort 140
** Gruppensumme **                             21040.40

** Lagerort 150
** Gruppensumme **                            112726.00

*** Gesamt ***                                154828.06
```

Bild 55 Liste "Wertmaessiger Lagerbestand", nach Lagerorten aufgeschluesselt, nur Gruppensummen und Summe (Reportdatei "best22")

Im folgenden Beispiel soll eine "Ersatzteiluebersicht" erstellt werden. Die Daten fuer diese Liste liefert die Datei "bestand", die wiederum indiziert wurde ("ortbest").

In der letzten Spalte der Liste soll der prozentuale Anteil des vorhandenen Bestandes am Mindestbestand ausgedruckt werden. Dazu wird die Berechnungsformel

```
anz/minbest*100
```

genommen. Es sollen jedoch nur gerundete ganzzahlige Prozentwerte ausgegeben werden. Um dies zu erreichen, wird zum errechneten Wert der Wert 0.5 addiert und von dieser Summe der ganzzahlige Anteil genommen. Dafuer dient die Funktion

```
INT (<ausdrN>)
```

(INT ist abgeleitet von INTEGER=ganze Zahl).

Bild 56 zeigt die Liste "Ersatzteiluebersicht", die mit

```
CREATE REPORT best31
```

erklaert wird und mit

```
REPORT FORM best31 TO PRINT
```

ausgedruckt wird.

Seite 1
09.09.89

Ersatzteiluebersicht

| Teile-Nr | Bezeichnung | Mindest-bestand | Bestand | Diff.-bestand | in % zum Mindest-bestand |
|---|---|---|---|---|---|
| ** Lagerort 110 | | | | | |
| 00031 | Sechskantmutter | 2400 | 5972 | 3572 | 249 |
| 00065 | Zylinderschraube | 1400 | 3472 | 2072 | 248 |
| 00106 | Federring | 2000 | 1374 | -626 | 69 |
| 00108 | Dichtring | 320 | 766 | 446 | 239 |
| 00852 | Bolzen | 600 | 471 | -129 | 79 |
| ** Lagerort 120 | | | | | |
| 00327 | Keilriemen | 120 | 233 | 113 | 194 |
| 00789 | Seilzug | 120 | 295 | 175 | 246 |
| 31100 | Kolben | 80 | 200 | 120 | 250 |
| 31200 | Keilriemenscheibe | 100 | 253 | 153 | 253 |
| ** Lagerort 130 | | | | | |
| 00353 | Scheinwerfer | 15 | 30 | 15 | 200 |
| 00730 | Lenkstock,vollst. | 25 | 60 | 35 | 240 |
| 17000 | Blinkleuchte,vorn | 50 | 114 | 64 | 228 |
| 30027 | Radzylinder | 20 | 6 | -14 | 30 |
| ** Lagerort 140 | | | | | |
| 00425 | Bremstrommel | 120 | 10 | -110 | 8 |
| 00819 | Bremsleitung | 150 | 31 | -119 | 21 |
| 14043 | Ansaugkruemmer | 80 | 194 | 114 | 242 |
| 30000 | Bremszylinder | 110 | 278 | 168 | 253 |
| 40359 | Kurbelwelle | 30 | 63 | 33 | 210 |
| ** Lagerort 150 | | | | | |
| 00516 | Batterie | 5 | 3 | -2 | 60 |
| 40018 | Motor | 30 | 79 | 49 | 263 |

Bild 56 Liste "Ersatzteiluebersicht" (Reportdatei "best31")

Fuer die Definition der Liste werden in den vom Reportgenerato bereitgestellten Menues folgende Eintragungen vorgenommen:

```
Format-Menue
------------
Ueberschrift:      Ersatzteiluebersicht
                   --------------------
Uebernahme der angebotenen Formatparameter


Gruppe-Menue
------------
Gruppenmerkmal:   ortnr
Gruppenueberschrift: Lagerort
keine weiteren Eintragungen


Spalte-Menue
------------
```

| Spalte | Inhalt | Kopf | Gesamt |
|---|---|---|---|
| 1 | teilnr | Teile-Nr | |
| 2 | bezei | Bezeichnung | |
| 3 | minbest | Mindest-bestand | N |
| 4 | anz | Bestand | N |
| 5 | anz-minbest | Diff.-bestand | N |
| 6 | INT(anz/minbest*100+0.5) | in % zum Mindest-bestand | N |

Man kann den Befehl REPORT FORM auch in Abhaengigkeit von Bedingungen formulieren, die wieder mit dem Zusatz FOR eingeleitet werden. Damit ist es moeglich, auch fuer die Formulierung einer Liste alle bekannten Selektionsmoeglichkeiten anzuwenden. Genauso ist es moeglich, Datensaetze mit SET FILTER (vgl. Abschnitte 3.2.3. und 3.9.) zu selektieren, bevor die Liste erstellt wird.

Es soll im folgenden - als letztes Beispiel der Anwendung des Reportgenerators - eine Bedarfsliste erstellt werden. Bedarf an Teilen besteht dann, wenn der tatsaechliche Bestand kleiner als der geforderte Mindestbestand ist. Ausgangsbasis dieser Liste ist die vorhandene Reportdatei "best31".

Diese Liste soll die zusaetzliche Ueberschrift "BEDARFSLISTE" tragen.
Der REPORT FORM-Befehl wird entsprechend formuliert:

```
REPORT FORM best31 FOR anz<minbest HEADING "BEDARFSLISTE"
```

Mit HEADING <zeichenreihe> wird eine zusaetzliche Ueberschrift ausgegeben.
Der REPORT FORM-Befehl setzt auch hier die eroeffnete Indexdatei "ortbest" voraus.

Bild 57 zeigt die "Bedarfsliste".

Seite 1 BEDARFSLISTE
09.09.89

Ersatzteiluebersicht

| Teile-Nr | Bezeichnung | Mindest-bestand | Bestand | Diff.-bestand | in % zum Mindest-bestand |
|---|---|---|---|---|---|
| ** Lagerort 110 | | | | | |
| 00106 | Federring | 2000 | 1374 | -626 | 69 |
| 00852 | Bolzen | 600 | 471 | -129 | 79 |
| ** Lagerort 130 | | | | | |
| 30027 | Radzylinder | 20 | 6 | -14 | 30 |
| ** Lagerort 140 | | | | | |
| 00425 | Bremstrommel | 120 | 10 | -110 | 8 |
| 00819 | Bremsleitung | 150 | 31 | -119 | 21 |
| ** Lagerort 150 | | | | | |
| 00516 | Batterie | 5 | 3 | -2 | 60 |

Bild 57 Bedarfsliste (Reportdatei "best31")

## 3.8. Etikettengenerator

Neben dem Unterprogramm zum Erstellen von Listen bietet REDABAS-4 ein komfortables Unterprogramm zum Ausdrucken von Adressenaufklebern, speziellen Kennzeichen fuer Artikel usw., den sogenannten Etikettengenerator.

Der Etikettengenerator arbeitet nach dem Prinzip menuegesteuerter Befehle (vgl. Abschn. 2.5.). Insbesondere sind die Unten- bzw. Oben-Pfeiltasten (oder <PgDn> bzw. <PgUp> zur Grobeinstellung) zur Fuehrung des Markierungsbalkens auf eine Option und anschliessend <ET> zur Auswahl dieser Option zu betaetigen. Fuer Eingaben gelten die vom Editieren bekannten Tastenkombinationen (vgl. Abschn. 3.1.3.). Zur Einstellung auf ein Menue sind die Rechts- bzw. Links-Pfeiltasten zu benutzen. Die auszuwertende Datei wird eroeffnet, und der Befehl zum Erstellen der Etikettendatei wird gegeben. Dieser Befehl lautet in seiner einfachen Form

```
CREATE LABEL <etikettendatei>
```

REDABAS-4 stellt daraufhin das Auswahl-Menue bereit. Es dient der Festlegung des Formates. REDABAS-4 bietet Parameter an, die beliebig ueberschrieben werden koennen. Im Inhalt-Menue wird der Inhalt des Etiketts definiert.
Das Ende-Menue gestattet die Auswahl von Speichern oder Abbruch.
REDABAS-4 vergibt automatisch den Dateityp .ETI fuer die <etikettendatei> und speichert diese Datei zur Erzeugung von Etiketten.

Der Befehl

```
LABEL FORM <etikettendatei>
```

eroeffnet die <etikettendatei> und bewirkt das Erzeugen der Etiketten.

Wir moechten Ihnen die sehr einfache Handhabung des Etikettengenerators wieder am Beispiel demonstrieren.

Es sollen fuer alle Kunden des Reparaturbetriebes (Datei "kunden") Adressenaufkleber ausgedruckt werden.
Geben Sie ein:

```
USE kunden
CREATE LABEL etikett
```

Auf dem Bildschirm erscheint das in Bild 58 dargestellte Auswahl-Menue des Etikettengenerators. (Sie koennen durch Betaetigen der <F1>-Taste die Hilfstafel mit den Tastenbelegungen zur Cursorbewegung im unteren Bildschirmteil zum Verloeschen bzw. zur Anzeige bringen.)

```
-----------------------------------------------------------------
| Auswahl                          Inhalt        Ende   11:03:27|
|-------------------------------------------                     |
|| Groesse definieren:   88,9 x 23,0 x 1   |                     |
||-----------------------------------------|                     |
|| Etikett-Breite:          35             |                     |
|| Etikett-Hoehe:            5             |                     |
|| Linker Rand:              0             |                     |
|| vert. Abstand (Zeilen):   1             |                     |
|| hor. Abstand (Stellen):   0             |                     |
|| Anzahl nebeneinander:     1             |                     |
|-------------------------------------------                     |
|                                                                |
|----------------------------------------------------------------|
||Cursor:<-- -->|Loe. Zeich.:Del|Einf. Zeil.:^N|Einf.:   Ins||
||Zeich.: <- -> |Loe. Wort  :^T |Hilfe      :F1|Zoom e:^PgDn||
||Wort: Home End|Loe. Zeile :^U |Abbruch:   ESC|Zoom a:^PgUp||
|----------------------------------------------------------------|
|                                                                |
|CREATE LABEL|<A:>|ETIKETT.ETI |Opt: 1/7                         |
|Markierungsbalken - ^v. Auswahl - <--'. Menueauswahl - <->.     |
|    Breite x Hoehe x Anzahl fuer vordefinierte Etiketts         |
-----------------------------------------------------------------
```

Bild 58 Auswahl-Menue des Etikettengenerators
(Auessere Gestaltung)

Fuer die auessere Gestaltung der Etiketten bietet REDABAS-4 die oben voreingestellten Werte.
Ausgehend von der definierten Laenge der auszugebenden Datenfelder und vom vorhandenen Papier (z.B. Etikettenpapier) koennen diese Werte ueberschrieben werden.

Wir moechten folgende Form fuer den Adressenausdruck waehlen:

```
Gisela Schmidt
Bahnhofstr. 43
THARANDT
8223
```

Als Etikettenbreite waehlen wir 25 Zeichen (Name 10 Zeichen, Vorname 8 Zeichen (lt. Struktur) mit Zwischenraum), fuer die Hoehe werden 4 Zeilen benoetigt.

Bei normalem A4-Druckerpapier koennen wir drei Etiketten nebeneinander unterbringen.

Wir waehlen fuer unser Beispiel also folgende Werte:

| | |
|---|---|
| Etikett-Breite: | 25 |
| Etikett-Hoehe: | 4 |
| hor. Abstand (Stellen): | 2 |
| Anzahl nebeneinander: | 3 |

Diese Werte werden ueberschrieben, die beiden anderen belassen.

Nach vollzogener Eingabe des letzten Wertes wird mit der Rechts-Pfeiltaste das Inhalt-Menue aufgerufen. Dort werden die Namen der Felder eingetragen, aus deren Inhalt sich der Etiketteninhalt zusammensetzt. Dazu wird die entsprechende Etikett-Zeile als Option ausgewaehlt und die Feldfolge entweder direkt eingegeben oder der Feldname aus der Liste der Datenfelder der aktivierten Datenbankdatei ausgewaehlt, die zuvor durch Betaetigen der <F10>-Taste bereitzustellen ist.

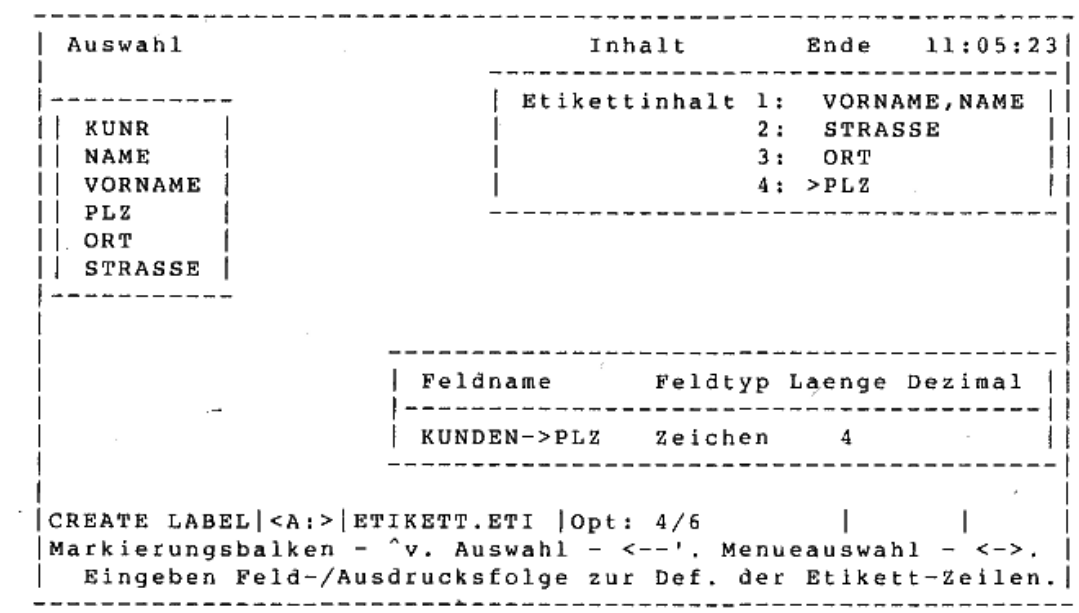



Bild 59 Inhalt-Menue des Etikettengenerators (Definition des Inhalts)

Bild 59 zeigt das Inhalt-Menue, nachdem die Hilfstafel mit <F1> abgestellt wurde und der letzte Feldnname "plz" ueber die Liste der Datenfelder der aktivierten Datenbankdatei einzubringen ist. ("plz" erscheint in der 4. Etikett-Zeile erst nach vollzogener Auswahl.) Genau die Zeilenanzahl, die im Auswahl-Menue unter Etikett-Hoehe eingetragen wurde, ist beschreibbar (d.h. auf dem Bildschirm hell eingerahmt).
Nehmen Sie die in Bild 59 dargestellten Eintragungen der Etikett-Zeilen vor!
Das Komma zwischen "vorname" und "name" bewirkt unabhaengig von der tatsaechlichen Laenge eines Vornamens jeweils eine Leerstelle zwischen Namen und Vornamen. (Das ist eine Besonderheit des Etikettengenerators, vgl. Befehlsbeschreibung in

Abschn. 9.3.3.1)
Waehlen Sie mit der Rechts-Pfeiltaste das Ende-Menue und darin die Option Speichern!

Mit dem Befehl

```
LABEL FORM etikett
```

werden die Etiketten erzeugt.

Wenn Sie den Drucker mit <CTRL-P> zugeschaltet haben oder den Zusatz TO PRINT angefuegt haben, werden die Etiketten in der in Bild 60 dargestellten Form ausgedruckt.

| | | |
|---|---|---|
| Gisela Schmidt<br>Bahnhofstr. 43<br>THARANDT<br>8223 | Heinz Schmidt<br>Waldweg 42<br>THARANDT<br>8223 | Franz Weber<br>Heideweg 66<br>THARANDT<br>8223 |
| Peter Lolle<br>Goethestr. 13<br>RADEBEUL<br>8122 | August Meier<br>Uferstr. 12<br>RADEBEUL<br>8122 | Inge Neumann<br>Gartenstr. 4<br>RABENAU<br>8222 |
| Karl Fleischer<br>Am See 83<br>FREITAL<br>8212 | Peter Miller<br>Nelkenweg 23<br>FREITAL<br>8212 | Otto Moser<br>Neugraben 8<br>FREITAL<br>8212 |
| Max Heinze<br>Rosenstr. 78<br>DRESDEN<br>8060 | Kurt Lehmann<br>Landstr. 15<br>DRESDEN<br>8060 | Ernst Schulze<br>Hauptstr. 61<br>DRESDEN<br>8060 |
| Fred Tauber<br>Fliederstr. 42<br>DRESDEN<br>8028 | | |

Bild 60 Ausdruck mit dem Etikettengenerator

Mit

```
MODIFY LABEL <etikettendatei>
```

sind jederzeit Veraenderungen bezueglich Inhalt und Form der Etiketten moeglich.

Sie koennen auch Probeausdrucke Ihres Etikettenformates veranlassen, in dem Sie mit

```
LABEL FORM <etikettendatei> SAMPLE
```

arbeiten. So koennen Sie gegebenenfalls vor dem Druck Korrekturen am Druckbild vornehmen.

## 3.9. Querygenerator

In Verbindung mit dem weiteren Ausbau menuegesteuerter Befehle in REDABAS-4 gegenueber seinen Vorgaengersystemen wird jetzt ein Querygenerator einbezogen, der Filterbedingungen generiert und den Umgang mit ihnen erleichtert. In Abschnitt 3.2.3. lernten Sie bereits das Definieren von Filterbedingungen fuer den SET FILTER TO -Befehl kennen. Bei einer neuen REDABAS-4-Sitzung oder Aenderung der Filterbedingung musste diese Filterbedingung in Verbindung mit dem SET FILTER TO -Befehl erneut eingetippt werden. Der Querygenerator bietet nun Hilfe beim Formulieren von bis zu 7 einfachen Filterbedingungen ( <feld><operator><ausdr> ) und beim Verbinden dieser mit logischen Operatoren <verbop.> und Klammern zu einer komplexen Filterbedingung, beispielsweise in der Form

```
<feld1><operator1><ausdr1><verbop.1>
(<feld2><operator2><ausdr2><verbop.2>
 <feld3><operator3><ausdr3>)
```

Das Ergebnis wird in einer Querydatei (Dateityp .QRY) gespeichert und ist jederzeit aufrufbar ueber den gegenueber Abschnitt 3.2.3. erweiterten Befehl

```
SET FILTER TO [FILE<querydatei>]/[<bedingung>]
```

Die Spezifikationen <querydatei> und <bedingung> sind gleichberechtigt. Nur muss eben eine <bedingung> von Hand eingegeben werden, waehrend fuer <querydatei> die Angabe des Dateinamens reicht.
Der Befehl zum Aufruf des Querygenerators lautet in vereinfachter Form

```
CREATE QUERY <querydatei>
```

Der Name fuer <querydatei> ist frei waehlbar. Falls keine Dateierweiterung spezifiziert wird, vergibt REDABAS-4 .QRY als Dateityp. Vor Aufruf des Querygenerators muss die gewuenschte Datenbankdatei (oder Viewdatei, vgl. Abschn. 3.11.4.) aktiviert sein, fuer die eine Filterbedingung aufgestellt werden soll.
Fuer die Arbeit mit dem Querygenerator sind die in Abschnitt 8.4.2. beschriebenen Tasten bzw. Tastenkombinationen gueltig. Insbesondere sind die Unten- bzw. Oben-Pfeiltasten (oder <PgDn> bzw. <PgUp> zur Grobeinstellung) zur Fuehrung des Markierungsbalkens auf eine Option und anschliessend <ET> zur Auswahl dieser Option zu betaetigen.

Es soll fuer die Datenbankdatei "kunden" eine Filterbedingung definiert werden, die nur die Familiennamen "Schmidt" akzeptiert, die in einem Wohnort mit Postleitzahlbeginn "821" oder "822" wohnen. Der Filter soll in der Querydatei "kundenql" gespeichert werden.
Geben Sie deshalb folgende Befehle ein:

```
USE kunden
CREATE QUERY kundenql
```

REDABAS-4 zeigt Ihnen das Setze-Filter-Menue an (vgl. Bild 61), in dem fuer die zu erstellende Querydatei "kundenql" alle 7 Zeilen der QUERY-Tabelle zur Aufnahme einfacher Filterbedingungen leer sind. Bei vorhandenen, zu modifizierenden Querydateien koennen in der QUERY-Tabelle alle Eintragungen ueber-

prueft und korrigiert werden. Durchlaufen Sie nun die einzelnen Optionen, um die QUERY-Tabelle zu fuellen! Waehlen Sie die Feldnamen-Option mit <ET> aus, und stellen Sie im Feldnamen-Untermenue auf NAME mit der Unten-Pfeiltaste ein! Die QUERY-Tabelle wird teilweise ueberlagert von der Anzeige der Feldattribute von NAME.

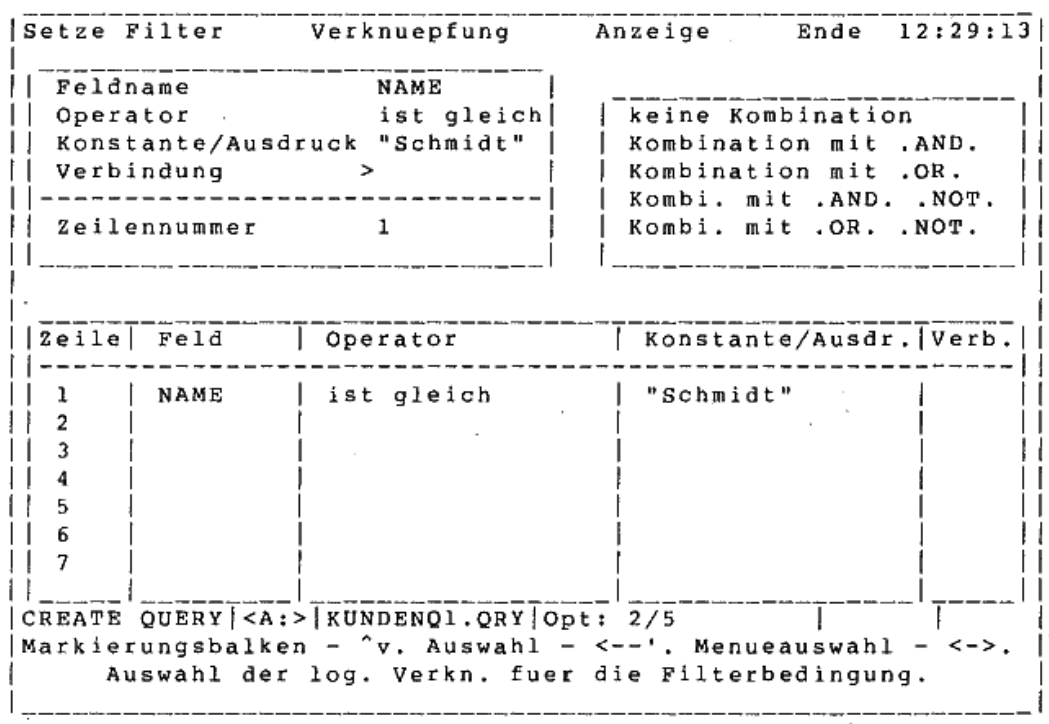



Bild 61 Setze-Filter-Menue des Querygenerators

Druecken Sie <ET>, erscheint der ausgewaehlte Feldname im Menue und in der QUERY-Tabelle, und der Markierungsbalken wandert auf die Operator-Option. Betaetigen Sie zweimal <ET>, um den "ist gleich"-Operator zu ueberfuehren. Waehlen Sie die Konstante/Ausdruck-Option mit <ET>, und geben Sie die Konstante "Schmidt" (mit Anfuehrungszeichen!) ein! (Bei der Formulierung von Ausdruecken mit Feldnamen duerfen Sie sich ueber <F10> das Feldnamen-Untermenue zu Hilfe nehmen.)
Bild 61 gibt den Zustand wieder, dass die Verbindung-Option mit <ET> ausgewaehlt wurde und der Markierungsbalken im Untermenue der Verbindungsoperatoren auf die 2. Option von 5 Optionen (vgl. Statuszeile) voreingestellt wurde. Druecken Sie <ET>! Die 1. Zeile der QUERY-Tabelle ist damit gefuellt, die Zeilennummer automatisch auf 2 erhoeht, und die Menuefelder sind fuer die Eingabe der 2. Zeile geleert.
Fahren Sie mit der Auswahl bzw. Eingabe fort, wie in der QUERY-Tabelle von Bild 62 angegeben!
Zur Kontrolle der definierten Filterbedingung springen Sie durch zweimaliges Betaetigen der Rechts-Pfeiltaste in das Anzeige-Menue. Die Filterbedingung ist formal in Ordnung, da ein Satz angezeigt wird. Durch Druecken von <PgDn> stellen Sie jedoch fest, dass auch andere Familiennamen als "Schmidt" auftauchen. Das bedeutet, wir muessen die Reihenfolge der Auswertung der einfachen Filterbedingungen (vgl. Abschnitt 8.3.3.3.)

durch Klammerung aendern. Dazu gehen wir mit der Links-Pfeiltaste in das Verknuepfung-Menue ueber. (Fuer Dateien mit mehr als 8 Feldern lohnt es sich, im Anzeige-Menue die <F1>-Taste zu betaetigen. Anstelle der QUERY-Tabelle werden bis zu 20 Datenfelder angezeigt. Bei Dateien mit mehr als 20 Datenfeldern oder bei angezeigter QUERY-Tabelle werden die jeweils naechsten Datenfelder eines Satzes mit <PgDn> ins Bild gerueckt.)

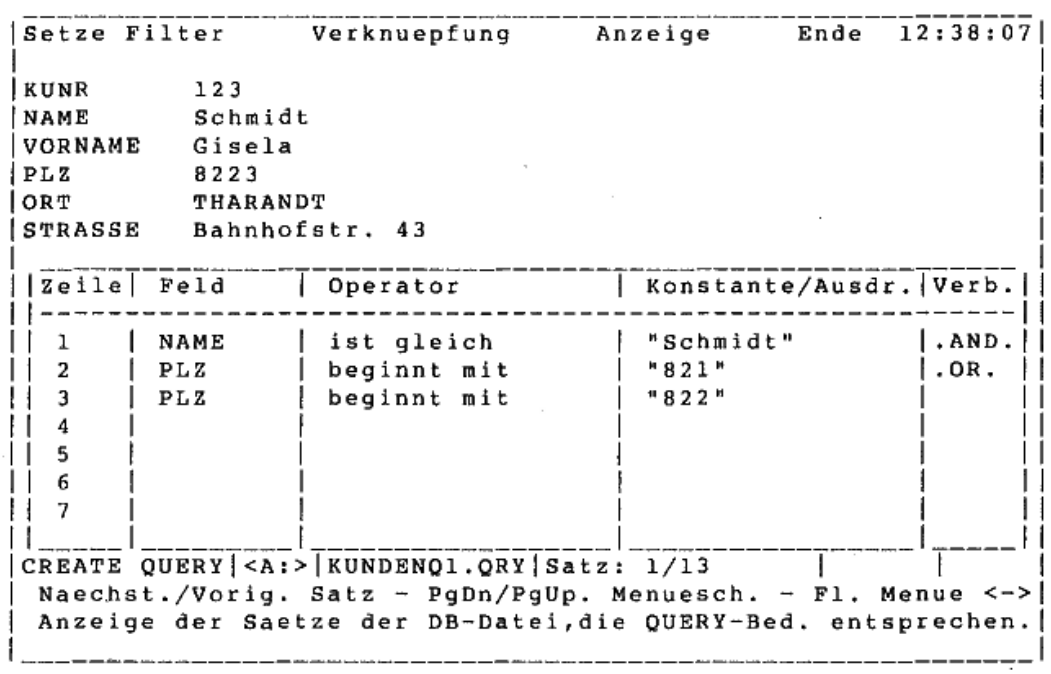



Bild 62 Anzeige-Menue des Querygenerators

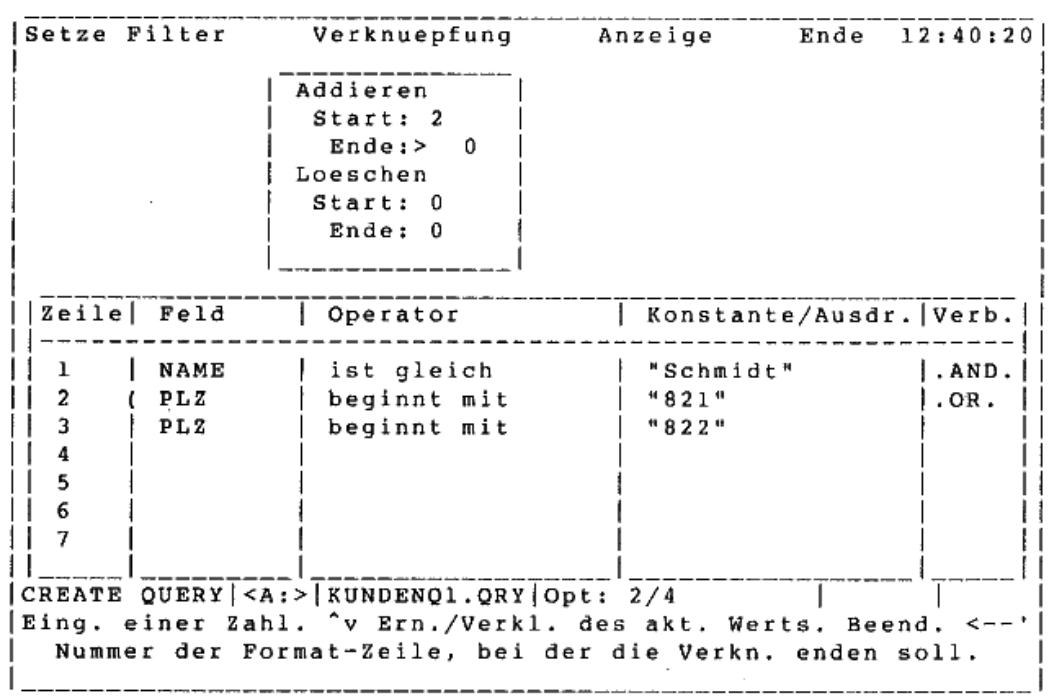



Bild 63 Verknuepfung-Menue des Querygenerators

Bild 63 zeigt das Verknuepfung-Menue, das die Klammerung bzw. Klammerloeschung der einfachen Filterbedingungen in der QUERY-Tabelle ermoeglicht. Die gestellte Aufgabe verlangt eine Klammerung der 2. und 3. Zeile der QUERY-Tabelle. Zum Setzen der linken oeffnenden Klammer in Zeile 2 waehlen Sie mit <ET> die Option Addieren Start und geben eine 2 ein (oder betaetigen zweimal die Oben-Pfeiltaste). Gehen Sie nun mit der Unten-Pfeiltaste zur Option Addieren Ende und waehlen Sie diese Option mit <ET> aus (s. Bild 63)! Geben Sie zum Setzen der rechten schliessenden Klammer in Zeile 3 eine 3 ein (oder betaetigen Sie dreimal die Oben-Pfeiltaste)!

Die Optionen Loeschen Start und Loeschen Ende eliminieren analog oeffnende und schliessende Klammern in den angegebenen Zeilen der QUERY-Tabelle.
Ueberzeugen Sie sich erneut im Anzeige-Menue von der Richtigkeit Ihrer definierten komplexen Filter-Bedingung! Es wirken nur solche einfachen Filterbedingungen der QUERY-Tabelle, die von der ersten Filterbedingung an mit einem Verbindungsoperator verbunden sind.
Gehen Sie mit der Rechts-Pfeiltaste in das Ende-Menue und waehlen Sie nach Wunsch die Optionen Speichern oder Abbruch! In beiden Faellen sind die vor Aufruf des Querygeneratorlaufs geoeffneten Dateien weiterhin geoeffnet. Im Falle der Option Speichern wird die Querydatei automatisch durch Ausgabe eines SET FILTER TO FILE -Befehls aktiviert und in dem angesteuerten Verzeichnis gesichert.
Sie koennen sich durch Betaetigen der <F6>-Taste (DISPLAY STATUS -Befehl) von der Aktivierung der Filterbedingung ueberzeugen. In unserem Beispiel ist zur Datei "kunden" folgende komplexe Filterbedingung durch REDABAS-4 automatisch erstellt worden (Die Bedeutung der Funktionen erfahren Sie im Abschn. 9.2.). Im Arbeitsbereich 1 wird ausgewiesen:

```
Filter : NAME = "Schmidt" .AND.
         (LEFT(TRIM(PLZ),LEN("821"))="821" .OR.
          LEFT(TRIM(PLZ),LEN("822"))="822")
```

Nachfolgende Befehle zu "kunden" verarbeiten nur Saetze, die dieser Filterbedingung genuegen, z.B. LIST.
Es ist moeglich, die Querydatei "kundenql" zu einem spaeteren Zeitpunkt mit dem Befehl

```
SET FILTER TO FILE kundenql
```

aufzurufen oder mit dem Befehl

```
MODIFY QUERY kundenql
```

zu modifizieren.

Im Falle der Option Abbruch werden die im gegenwaertigen Querygeneratorlauf vollzogenen Generierungen verworfen. Der Abbruch kann auch aus den Hauptmenues des Viewgenerators mit <Esc> erzielt werden. REDABAS-4 fragt sicherheitshalber, ob tatsaechlich abgebrochen werden soll.

## 3.10. Maskengenerator

Was fuer das Vorgaengersystem REDABAS als Zusatzkomponente unter dem Namen REDAMASK vom Kombinat Robotron angeboten wird, ist in aehnlicher Weise in REDABAS-4 als Maskengenerator integriert.

Der Maskengenerator ist ein seitenorientiertes, menuegesteuertes Unterprogramm zur Erstellung von individuellen Ein-/Ausgabemasken. Bis jetzt haben Sie gemaess Abschnitt 3.1. nur mit einer standardmaessigen Ein-/Ausgabemaske (vgl. Bild 6) fuer den Bildschirm in Verbindung mit den Befehlen CREATE, APPEND, INSERT, EDIT und CHANGE gearbeitet. Es ist jedoch fuer die vier zuletzt genannten Befehle moeglich, anstelle der standardmaessigen eine individuelle Ein-/Ausgabemaske aufzurufen, in der die Maske entsprechend Ihren Wuenschen benutzerfreundlicher gestaltet werden kann (vgl. Bild 64).

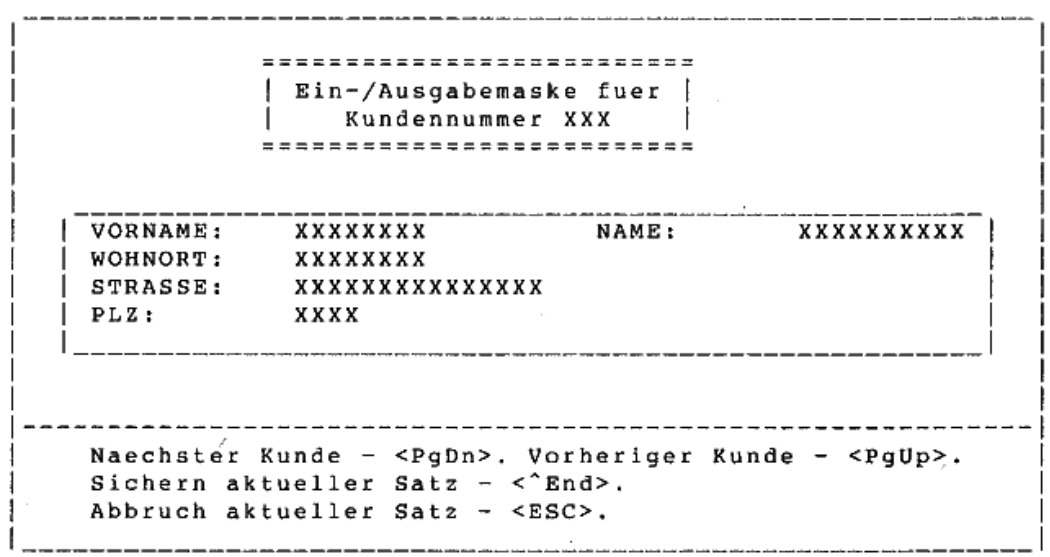



Bild 64 Individuelle Ein-/Ausgabemaske fuer Datenbankdatei "kunden"

Der Maskengenerator entbindet Sie beim Erstellen oder Aendern einer Maske von der Niederschrift entsprechender @- und READ-Befehle (vgl. Abschnitte 4.1. und 4.5.), vom Auszaehlen der Bildschirmkoordinaten und vom direkten Eingeben der Feldnamen, -masken und Grafik (Einfach- und Doppellinien).

Ueber das Entwerfen einer individuellen Ein-/Ausgabemaske hinaus erlaubt der Maskengenerator die Erstellung oder Aenderung der Struktur einer zugehoerigen Datenbankdatei und die Ausgabe einer Textdatei zur Dokumentation der Ein-/Ausgabemaske. Das Ergebnis des Maskengenerators wird sowohl in einer Screendatei (Dateityp .SCR) als auch in einer Maskendatei (Dateityp .MSK) gespeichert und ist damit zu einem spaeteren Zeitpunkt aufrufbar. Waehrend die Screendatei fuer nachfolgende Aenderungen der Ein-/Ausgabemaske mit Hilfe des Maskengenerators notwendig ist, dient die Maskendatei zur Bereitstellung der Ein-/Ausgabemaske.

Der Befehl zum Aufruf des Maskengenerators lautet in vereinfachter Form

```
CREATE SCREEN <screendatei>
```

Der Name fuer <screendatei> kann beliebig vorgegeben werden. Falls keine Dateierweiterung spezifiziert wird, fuegt REDABAS-4 den Dateityp .SCR an. Die gleichzeitig entstehende Maskendatei erhaelt den Namen der <screendatei> mit dem nicht anders vorgebbaren Dateityp .MSK. Eine Datenbankdatei, fuer die eine eigene Ein-/Ausgabemaske erstellt werden soll, braucht nicht unbedingt vor Aufruf des Maskengenerators eroeffnet zu sein (im Gegensatz zu einer Viewdatei).

Da der Maskengenerator nach dem Prinzip menuegesteuerter Befehle arbeitet (vgl. Abschnitt 2.5.), sind die im Abschnitt 8.4.2. beschriebenen Tasten bzw. Tastenkombinationen anzuwenden. Insbesondere sind die Unten- bzw. Oben-Pfeiltasten (oder

<PgDn> bzw. <PgUp> als Grobeinstellung) zur Fuehrung des Markierungsbalkens auf eine Option und anschliessend <ET> zur Auswahl dieser Option zu betaetigen. Zusaetzlich zu dem bereits bekannten Prinzip der Menuesteuerung im Report-, Etiketten- und Querygenerator wird im Maskengenerator mit dem sogenannten "Zeichenbrett" gearbeitet, einer Darstellung der zu entwerfenden individuellen Ein-/Ausgabemaske auf dem Bildschirm unter Steuerung eines Bildschirmeditors. Das Zeichenbrett dient zum Editieren von Text, Feldern und der Grafik in der Ein-/Ausgabemaske. Die Ein-/Ausgabemaske erscheint bei spaeterer Verwendung in der Form und Wirkung, wie sie im Zeichenbrett festgelegt wurde. Das wechselseitige Umschalten zwischen den Menues des Maskengenerators und dem Zeichenbrett erfolgt mit der <F10>-Taste.

Zur Demonstration des Maskengenerators soll die in Bild 64 angegebene Ein-/Ausgabemaske fuer die Datenbankdatei "kunden" zum Zwecke des Editierens erstellt werden. Die Kundennummer eines Satzes soll dabei nur angezeigt, aber nicht veraendert werden. Der Name der Screen- bzw. Maskendatei sei "kundensl.scr" bzw. "kundensl.msk".
Eine detaillierte Beschreibung finden Sie unter der Ueberschrift "Arbeit mit dem Maskengenerator fuer den Aufbau der Screen- bzw. Maskendatei "kundensl"".
Wer ohne viel Nachschlagen und ohne ausfuehrliche Kommentare die Ein-/Ausgabemaske von Bild 64 erzeugen moechte, kann die in Bild 65 angegebenen Aktionen ausfuehren, und anschliessend den unter der Ueberschrift "Arbeit mit der Masken- bzw. Screendatei "kundensl"" beschriebenen Umgang mit den erzeugten Dateien vornehmen.

Zunaechst sind die Aktionen in der linken Haelfte des Bildes, danach die der rechten Haelfte auszufuehren. Mit Ausnahme von <PgDn>, <PgUp>, <^End> und <Esc> sind die in den Textzeilen angegebenen Tasten zu betaetigen. Nach <ET> aufgefuehrter Text beinhaltet eine eingestellte Option eines Menues bzw. Untermenues oder einen Kommentar. Der restliche Text (Befehle und Text der Ein-/Ausgabemaske) ist einzugeben.

## Arbeit mit dem Maskengenerator fuer den Aufbau der Screen- bzw. Maskendatei "kundensl"

Bevor Sie den Maskengenerator aufrufen, sollen sicherheitshalber alle Dateien geschlossen werden. Das erreichen Sie mit dem Befehl

```
CLOSE ALL   bzw.   CLOSE DATABASES
```

Andernfalls wird eine Maske fuer die im selektierten Arbeitsbereich eroeffnete Datenbank- oder fuer eine aktivierte Viewdatei erstellt.
Geben Sie den Befehl zum Aufruf des Maskengenerators ein:

```
CREATE SCREEN kundensl
```

Auf dem Bildschirm wird das Maskengenerator-Menue mit dem voreingestellten Aufbau-Menue angezeigt. Druecken Sie <ET> zur Auswahl der Datenbankdatei-auswaehlen-Option!

```
CLOSE DATABASES<ET>
CREATE SCREEN kundensl<ET>
<ET> Datenbankdatei auswaehl.
<ET> KUNDEN.DBD
<F10>
7x <Unten-Pfeiltaste>
6x <->>
<F10>
<<->
<Oben-Pfeiltaste>
<ET> Felder laden
<Unten-Pfeiltaste>
<ET> NAME
<Unten-Pfeiltaste>
<ET> VORNAME
<Unten-Pfeiltaste>
<ET> PLZ
<Unten-Pfeiltaste>
<ET> ORT
<Unten-Pfeiltaste>
<ET> STRASSE
<<->
<Ins>
30x <Leertaste>
<Ins>
4x <->>
:
35x <<->
VORNAME:
4x <->>
<Unten-Pfeiltaste>
<ET> Anfang der Verschiebung
<Oben-Pfeiltaste>
<ET> Ende der Verschiebung
<Unten-Pfeiltaste>
<CTRL-Y>
<ET> Anfang des Verschiebens
3x <Unten-Pfeiltaste>
<ET> Ende des Verschiebens
12x <<->
PLZ:
3x <Oben-Pfeiltaste>
<CTRL-Y>
4x <<->
WOHNORT:
<Unten-Pfeiltaste>
<<->
:
7x <Oben-Pfeiltaste>
4x <->>
Ein-/Ausgabemaske fuer
<Unten-Pfeiltaste>
19x <<->
Kundennummer
<Leertaste>
<F10>
<Unten-Pfeiltaste>
<ET> Feld
<Unten-Pfeiltaste>
<ET> KUNR
<Oben-Pfeiltaste>
<ET> Edit/GET
<F10>
<F10>
<->>
<Oben-Pfeiltaste>
<ET> Doppelte Linie
18x <<->
<Unten-Pfeiltaste>
<ET> Rahmenanfang
3x <Oben-Pfeiltaste>
25x <->>
<ET> Rahmenende
<F10>
<->>
<Unten-Pfeiltaste>
<ET> Enfache Linie
5x <Unten-Pfeiltaste>
37x <<->
<ET> Rahmenanfang
5x <Unten-Pfeiltaste>
55x <->>
<ET> Rahmenende
<F10>
<->>
<Unten-Pfeiltaste>
<ET> Einfache Linie
3x <Unten-Pfeiltaste>
2x <->>
<ET> Streckenanfang
59x <<->
<ET> Streckenende
<Unten-Pfeiltaste>
4x <->>
Naechster Kunde - <PgDn>.
Vorheriger Kunde - <PgUp>.<ET>
6x <->>
Sichern aktueller Satz - <^End>
.<ET>
6x <->>
Abbruch aktueller Satz - <ESC>.
<F10>
<->>
<ET> Generieren Textdatei
<->>
<ET> Speichern
```

Bild 65 Aktionen zum Aufbau der Maskendatei "kundensl.msk"

Es wird eine Liste der auf dem aktuellen Laufwerk vorhandenen Datenbankdateien (siehe Bild 66) als Untermenue angeboten. Waehlen Sie "KUNDEN.DBD" aus! Springen Sie mit <F10> ins Zeichenbrett! Das Zeichenbrett ist zur Zeit leer.

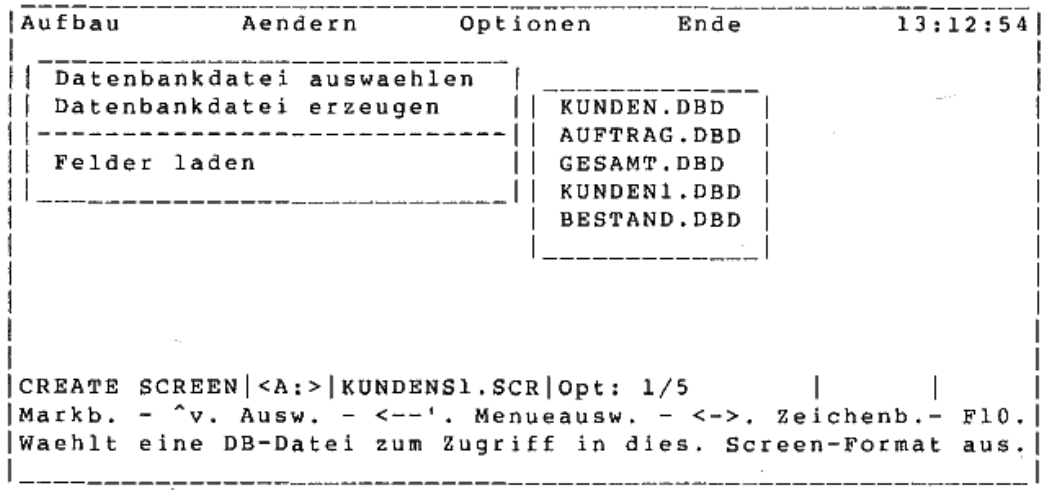



Bild 66 Aufbau-Menue des Maskengenerators

Zur Vorbereitung des Einbringens von Feldnamen und Feldmasken positionieren Sie den Cursor an die Stelle, wo der Feldname "VORNAME" beginnen soll (Row 07, Col 06). Die Positionsangaben des Cursors werden in der Statuszeile durch Seite (Pg), Zeile (Row) und Spalte (Col) ausgewiesen. Zeilen- und Spaltennummer werden von Null an gezaehlt.
Mit <F10> kommen Sie in das Aendern-Menue. Die Betaetigung der Links- und Oben-Pfeiltaste fuehrt zur Felder-laden-Option des Aufbau-Menues. Waehlen Sie diese Option aus!

Anstelle der Datenbankdateinamen in Bild 66 wird eine Liste der Feldnamen von "kunden" aufgerufen. Uebergehen Sie das Feld KUNR und waehlen Sie die restlichen Felder aus. Die ausgewaehlten Felder muessen alle durch einen Pfeil gekennzeichnet sein.
Mit der Links-Pfeiltaste springen Sie ins Zeichenbrett.

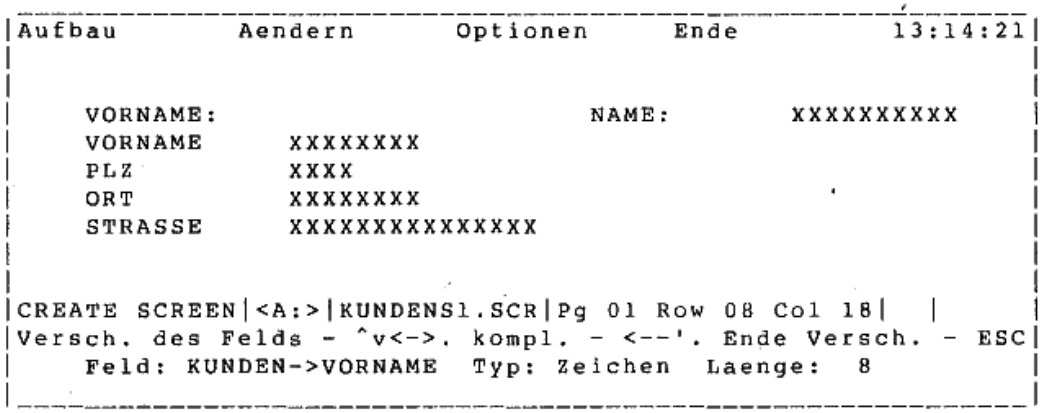



Bild 67 Zeichenbrett des Maskengenerators

Sie koennen sich davon ueberzeugen, dass Feldnamen und Feldmasken untereinander in der Reihenfolge des Auftretens in der Liste der Feldnamen eingebracht sind. Die Felder muessen nun entsprechend Bild 64 verschoben werden. Stellen Sie mit <Ins> den Einfuegemodus ein und betaetigen Sie mehrmals die

Leertaste! Das Feld NAME verschiebt sich mit der Feldmaske. Schalten Sie mit <Ins> den Einfuegemodus aus und fuegen Sie den Doppelpunkt nach NAME ein! Tragen Sie gemaess Bild 67 VORNAME: in die gleiche Zeile ein!. Bewegen Sie den Cursor auf den Anfang der Feldmaske von VORNAME und druecken Sie <ET> zum Zwecke des Verschiebens der Feldmaske eine Zeile nach oben! Bild 67 gibt diesen Zustand wieder.

Der Cursor ist um eine Zeile nach oben zu bewegen und <ET> zu druecken. Diese Zeile entspricht nun der Vorgabe aus Bild 64. Stellen Sie den Cursor eine Zeile nach unten und loeschen Sie die Zeile mir <CTRL-Y>. Auf analoge Art ist die Feldmaske von PLZ zu verschieben, die neue Zeile um PLZ: zu ergaenzen und die alte Zeile mit <CTRL-Y> zu loeschen. Der Feldname ORT soll in der Ein-/Ausgabemaske in WOHNORT: geaendert werden. Die Feldnamen zum Editieren in der Ein-/Ausgabemaske sollen jeweils mit einem Doppelpunkt abgeschlossen werden.
Nachdem Sie den Text fuer die zwei Zeilen der Ueberschrift eingegeben haben (s. Bild 64), holen Sie die Feldmaske fuer KUNR ueber das Aendern-Menue. Der Feldinhalt von KUNR sollte uebrigens nur angezeigt werden und fuer Eingaben gesperrt sein. Positionieren Sie den Cursor auf das 1. Zeichen der Feldmaske von "Kundennummer" und druecken Sie <F10>! Es erscheint das Aendern-Menue (siehe Bild 68).

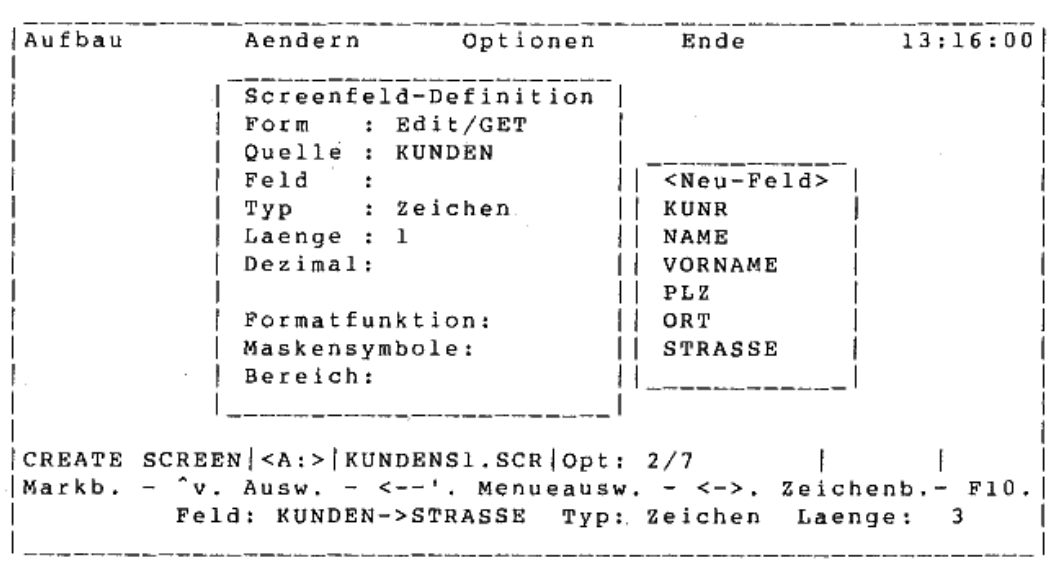



Bild 68 Aendern-Menue des Maskengenerators

Waehlen Sie die Option Feld und aus dem Untermenue den Feldnamen KUNR! Waehlen Sie die Option Form, um Edit/GET in Display/SAY zu aendern, und betaetigen Sie <F10>! Die Feldmaske zu KUNR ist damit ins Zeichenbrett gebracht. Sie ist im Gegensatz zu den anderen nicht hell unterlegt. Das bedeutet, dass Sie fuer Eingaben gesperrt ist.

Als naechste Aktion ist die Grafik ins Zeichenbrett zu bringen. (Erinnern Sie sich daran, dass Sie durch Betaetigen von <F10> das Zeichenbrett "ein- oder ausschalten".) Waehlen Sie das Optionen-Menue und darin die Option Doppelte Linie (siehe Bild 69)! Nach der Auswahl sind Sie im Zeichenbrett. Bewegen

Sie den Cursor auf den linken unteren Eckpunkt des zu zeichnenden Doppelrahmens mit <ET> als Abschluss! Fuehren Sie den Cursor analog auf den rechten oberen Eckpunkt, und druecken Sie <ET>! Der Doppelrahmen von Bild 64 erscheint im Zeichenbrett. Nachdem Sie die Ein-/Ausgabemaske durch Zeichnen des einfachen Rahmens und der Eingabe des im unteren Teil von Bild 64 dargestellten Textes vervollstaendigt haben, waehlen Sie die Option Generieren einer Textdatei im Optionen-Menue. Anschliessend Waehlen Sie die Option Speichern im Ende-Menue.

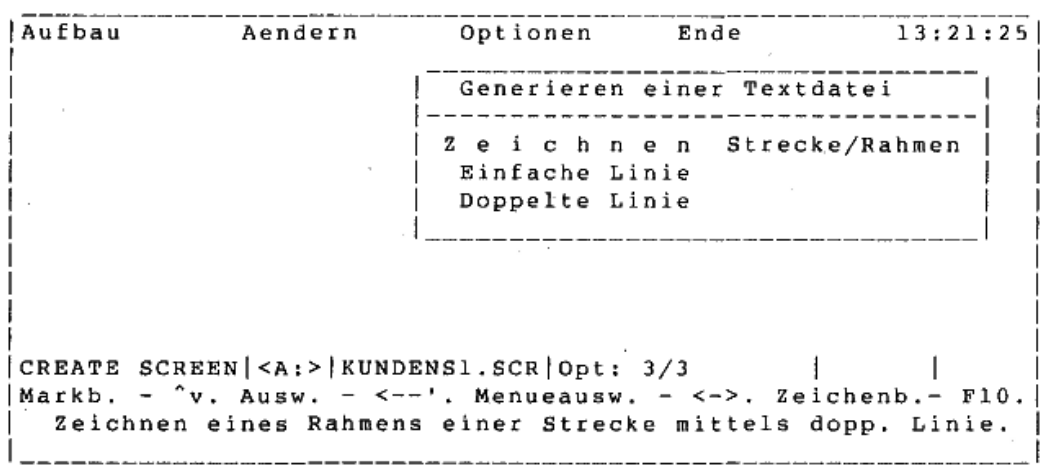



Bild 69 Optionen-Menue des Maskengenerators

Damit wird die Screen- und Maskendatei - in unserem Fall auch die Textdatei - in dem angesteuerten Verzeichnis gesichert. Die Maskendatei wird automatisch durch Ausgabe eines

SET FORMAT TO <maskendatei> -Befehls

aktiviert.
Sie koennen jederzeit aus den Hauptmenues des Maskengenerators mit <Esc> oder ueber die Option Abbruch des Ende-Menues den Lauf abbrechen. Die bis dahin vorgenommenen Aktionen werden ignoriert. REDABAS-4 fragt sicherheitshalber, ob abgebrochen oder weitergearbeitet werden soll.

## Arbeit mit der Masken- bzw. Screendatei "kundens1"

Nachdem Sie den Maskengenerator ueber die Option Speichern verlassen haben, koennen Sie sich durch Betaetigen der <F6>-Taste (DISPLAY STATUS -Befehl) von der eroeffneten Maskendatei ueberzeugen. In unserem Beispiel ist zur Datei "kunden" unter Arbeitsbereich 1 vermerkt:

Maskendatei: A:kundens1.msk

Kontrollieren Sie durch Aufruf des EDIT-, APPEND-, CHANGE- oder INSERT-Befehls, dass Ihre individuelle Ein-/Ausgabemaske auf dem Bildschirm erscheint! Vereinbarungsgemaess bleibt die Feldmaske fuer die Kundennummer bei EDIT gesperrt.
Wenn Sie die Maskendatei mit dem Befehl

```
TYPE kundensl.msk TO PRINT
```

ausdrucken, erhalten Sie eine Liste gemaess Bild 70.

Fuer Zwecke der Dokumentation ist es ratsam, eine im Optionen-Menue generierte Textdatei mit dem Befehl TYPE aufzulisten, z.B.

```
TYPE kundensl.txt TO PRINT
```

Das Ergebnis ist eine Liste entsprechend Bild 71 und eine Darstellung der Ein-/Ausgabemaske, jedoch ohne Grafik, entsprechend Bild 64.
Es ist moeglich, die Maskendatei "kundensl.msk" zu einem spaeteren Zeitpunkt mit dem Befehl

```
SET FORMAT TO kundensl
```

aufzurufen oder sie ueber die Screendatei "kundens.scr" mit dem Befehl

```
MODIFY SCREEN kundensl
```

zu modifizieren.

```
@  2,18 SAY "Ein-/Ausgabemaske fuer"
@  3,21 SAY "Kundennummer"
@  3,34 SAY  KUNDEN->KUNR
@  7, 6 SAY "VORNAME:"
@  7,18 GET  KUNDEN->VORNAME
@  7,36 SAY "NAME:"
@  7,48 GET  KUNDEN->NAME
@  8, 6 SAY "WOHNORT:"
@  8,18 GET  KUNDEN->ORT
@  9, 6 SAY "STRASSE:"
@  9,18 GET  KUNDEN->STRASSE
@ 10, 6 SAY "PLZ:"
@ 10,18 GET  KUNDEN->PLZ
@ 15, 6 SAY "Naech. Kunde - <PgDn>. Vorh. Kunde - <PgUp>."
@ 16, 6 SAY "Sichern aktueller Satz - <^END>."
@ 17, 6 SAY "Abbruch aktueller Satz - <ESC>."
@  1,16 TO  4,41     DOUBLE
@  6, 4 TO 11,59
@ 14, 2 TO 14,61
```

Bild 70  Ausdruck der erzeugten Maskendatei "kundensl.msk"

```
Felddefinition fuer das Format : kundensl.scr

Seite Row Col DB-Datei Feld       Typ      Laenge  Dez
  1    7   48  KUNDEN   NAME      Zeichen      10
  1    7   18  KUNDEN   VORNAME   Zeichen       8
  1   10   18  KUNDEN   PLZ       Zeichen       4
  1    8   18  KUNDEN   ORT       Zeichen       8
  1    9   18  KUNDEN   STRASSE   Zeichen      15
  1    3   34  KUNDEN   KUNR      Zeichen       3
```

Bild 71  Teil des Ausdrucks der Textdatei "kundensl.txt"

## 3.11. Gleichzeitiges Arbeiten mit mehreren Datenbankdateien

Redundanzarmes Speichern, das zu den Schwerpunkten der Datenbanktechnologie gehoert, erfordert einen flexiblen Datenbankzugriff. So kommt es darauf an, Daten, die nur einmal in der Datenbank gespeichert sind, von unterschiedlichen Programmen aus anzusprechen und zu bearbeiten und sie gegebenenfalls aus verschiedenen Dateien zusammenzufuehren.

Mit REDABAS-4 koennen bis zu 10 verschiedene Datenbankdateien in 10 Arbeitsbereichen gleichzeitig geoeffnet sein. Die Dateien koennen unabhaengig voneinander bearbeitet werden. Bearbeitet man eine Datei (die sogenannte aktuelle Datei), kann der in den anderen geoeffneten Dateien jeweils aktuelle Satz gelesen, aber nicht veraendert werden.

Fuer das gleichzeitige Bearbeiten von mehreren Dateien und das Erzeugen einer neuen Datei aus zwei bestehenden bietet REDABAS-4 als relationales Datenbankbetriebssystem sehr leistungsfaehige Befehle, die nachfolgend vorgestellt werden.

### 3.11.1. Arbeitsbereiche und Aliasnamen

REDABAS-4-Dateien werden in Arbeitsbereichen eroeffnet. Entsprechend der Moeglichkeit des gleichzeitigen Zugriffs auf bis zu 10 Dateien sind die Arbeitsbereiche von 1 bis 10 (bzw. A bis J) durchnumeriert.

Beim Aufruf von REDABAS-4 befinden Sie sich immer im Arbeitsbereich 1. Wollen Sie jetzt eine weitere Datei oeffnen - ohne dass die erste geschlossen wird - muessen Sie einen anderen Arbeitsbereich auswaehlen und in diesem die Datei eroeffnen. Der Befehl, mit dem Arbeitsbereiche ausgewaehlt werden, lautet

SELECT

Sie koennen mit diesem Befehl in den 10 moeglichen Arbeitsbereichen beliebig umschalten, das heisst 10 Dateien auswaehlen, ohne dass die Datei, die Sie verlassen, geschlossen wird.

Haben Sie beispielsweise in Datei 1 den 10. Datensatz bearbeitet und wechseln Sie mit SELECT in eine zweite Datei, um den 20. Datensatz auszuwerten, dann ist, wenn Sie mit SELECT zur ersten Datei zurueckkehren, dort immer noch Datensatz 10 aktuell.
Kommen Sie wieder zur zweiten Datei, ist dort der 20. Datensatz noch aktiviert. (Beachten Sie den Unterschied zur Eroeffnung einer Datenbank mit USE, wo stets der erste Datensatz aktiviert wird!)

Jedes Mal, wenn Felder einer Datei aus einem anderen Bereich angesprochen werden, muss die Dateizugehoerigkeit erklaert werden. Damit die mitunter recht langen Dateinamen nicht immer voll ausgeschrieben werden muessen, erlaubt REDABAS-4 die Vergabe von Aliasnamen.
Der Aliasname wird beim Eroeffnen der Datenbankdatei mit USE spezifiziert. Bis zum Schliessen der Datenbankdatei kann sie

ueber diesen bei USE spezifizierten Aliasnamen angesprochen werden, beim Schliessen geht der Aliasname verloren.

Fuer die Vergabe von Aliasnamen gelten die gleichen Regeln wie fuer Feldnamen (vgl. Abschn. 3.1.1.).
Wird innerhalb eines Befehles der Aliasname verwendet, z. B. bei der Arbeit mit mehreren Dateien zur Zuordnung von Datenfeldern, wird der Aliasname dem Feldnamen vorangestellt und Alias- und Feldname werden mit einem Pfeil (zusammengesetzt aus - und >) verbunden.

### 3.11.2. Gleichzeitiges Bearbeiten von Datenbankdateien mit unterschiedlicher Dateistruktur

Bevor diese Moeglichkeit demonstriert wird, soll eine neue Datenbankdatei mit dem Namen "aufpos" erstellt werden. Diese Datei soll saemtliche Positionen pro Auftrag erfassen und soll die Felder "aufnr" (Auftragsnummer), "teilnr" (Teilnummer), "anz" (Anzahl) und "betrag" enthalten. Die Struktur der Datei ist in Bild 72 dargestellt, die Datei selbst in Bild 73.

Datenbankdateistruktur : A:aufpos.dbd
Anzahl der Datensaetze : 5
Letztes Aenderungsdatum: 05.10.89

| Feld | Feldname | Typ | Laenge | Dez |
|---|---|---|---|---|
| 1 | AUFNR | Zeichen | 4 | |
| 2 | TEILNR | Zeichen | 5 | |
| 3 | ANZ | Numerisch | 4 | |
| 4 | BETRAG | Numerisch | 8 | 2 |
| ** Gesamt ** | | | 22 | |

Bild 72 Struktur der Datenbankdatei "aufpos"

| Satznr. | AUFNR | TEILNR | ANZ | BETRAG |
|---|---|---|---|---|
| 1 | 4567 | 40359 | 1 | 0.00 |
| 2 | 4567 | 14043 | 1 | 0.00 |
| 3 | 4567 | 00106 | 4 | 0.00 |
| 4 | 4567 | 00031 | 4 | 0.00 |
| 5 | 8423 | 40018 | 1 | 0.00 |

Bild 73 Datenbankdatei "aufpos"

Sie erinnern sich sicher, dass in der Datenbankdatei "bestand" (s. Abschnitt 3.7.) u. a. die Information ueber den Preis eines Teiles enthalten ist. Um die Kosten der Teile eines Auftrages und dessen Gesamtkosten zu ermitteln, ist es also erforderlich, aus dieser Bestandsdatei den Einzelpreis pro Teil zu entnehmen, aus der Datei "aufpos" die Anzahl der Teile pro Auftrag und beide Werte miteinander zu multiplizieren.
Die Datei "bestand" und die Datei "aufpos" muessen gleichzeitig im Zugriff sein.

Wir eroeffnen dazu im Arbeitsbereich 1 die Datei "aufpos" mit dem Aliasnamen "auf" und im Arbeitsbereich 2 die Datei "bestand" mit dem Aliasnamen "be" und der zugehoerigen Indexdatei "teilbes", die vorher auf bekannte Weise mit "teilnr" als Schluessel erstellt wurde. (Dies erfordert der spaeter folgende SET RELATION-Befehl.)

```
SELECT 1
USE aufpos ALIAS auf

SELECT 2
USE bestand INDEX teilbes ALIAS be
```

Beide Dateien besitzen gemeinsam das Datenfeld "teilnr", ueber das sie korrespondieren. Um aus beiden Dateien Datensaetze mit gleicher Teilnummer bearbeiten zu koennen, muessen die Dateien bezueglich des genannten Feldes "teilnr" synchronisiert werden. Dazu dient der Befehl

```
SET RELATION TO <feld> INTO <alias>
```

als Variante des im Abschn. 9.3.3. beschriebenen SET RELATION-Befehls. Im Abschnitt 3.11.1. erfuhren Sie, dass die Datensatzzeiger der in den einzelnen Arbeitsbereichen eroeffneten Dateien voneinander unabhaengig sind. Das heisst, wird der Datensatzzeiger einer Datei veraendert, hat das auf den Datensatzzeiger anderer geoeffneter Dateien keinen Einfluss.
Will man die Datensatzzeiger der aktuellen und einer zweiten geoeffneten Datenbankdatei synchronisieren, muss der SET RELATION-Befehl gegeben werden. Beide Dateien werden durch diesen Befehl zueinander in "Beziehung" (Relation) gebracht.

Der Datensatzzeiger der aktuellen Datei steuert dabei den Datensatzzeiger der Datei, die in dem SET RELATION-Befehl genannt wird. Diese Synchronisation kann z.B. anhand der Datensatznummer oder eines gemeinsamen Feldes, nach dem die zu synchronisierende Datei indiziert sein muss, erfolgen (vgl. Abschn. 9.3.3.).

Auf diese Weise koennen bis zu 10 Dateien, die korrespondierende Informationen enthalten, aneinander gebunden werden.

In unserem Beispiel soll die Datei "bestand" (Aliasname be) an die Datei "aufpos" gebunden werden, das Verbindungselement ist dabei die Teilnummer.

Die Befehle

```
SELECT 1
SET RELATION TO teilnr INTO be
```

realisieren nun, dass beispielsweise dann, wenn in der Datei "aufpos" der Datensatz mit der Teilnummer 14043 bearbeitet wird, in der synchronisierten Datei "bestand" automatisch der Datensatz mit der gleichen Teilnummer bereitgehalten wird. Fuer die Bearbeitung dieses Satzes muesste dann allerdings der Arbeitsbereich gewechselt werden oder die Feldnamen ueber den Aliasnamen angesprochen werden. In Abschnitt 3.11.4. wird beschrieben, wie Sie SET RELATION-Befehle und die notwendigen Datenbankdateieroeffnungen mit Hilfe des Viewgenerators generieren koennen.

Sie erinnern sich sicher, dass das Ziel unseres Beispiels darin besteht, die Kosten der Teile eines Auftrags und dessen Gesamtkosten zu ermitteln, d. h. das Feld "betrag" der Datei "aufpos", das gegenwaertig 0.00 enthaelt (vgl. Bild 73) durch den sich aus preis*anz ("preis" aus Datei "bestand", "anz" aus Datei "aufpos") ergebenden Betrag zu ersetzen.

Wir verwenden dafuer den schon im Abschnitt 3.1.3. erwaehnten REPLACE-Befehl.

```
REPLACE betrag WITH be->preis * anz
```

Beachten Sie, dass Feldern, die ausserhalb des aktuellen Bereiches angesprochen werden, der Aliasname vorangestellt werden muss. Da wir uns im ersten Arbeitsbereich befinden, gilt dies fuer das Datenfeld "preis" der Datei "bestand".

Nachdem die Kosten der ersten Position eines Auftrags berechnet wurden, gehen wir zur naechsten Auftragsposition ueber. Der naechste Satz der Datei im Bereich 1 ("aufpos") muss auf die gleiche Weise bearbeitet werden.
Mit dem Befehl

```
SKIP
```

wird zum naechsten Datensatz uebergegangen und der gleiche REPLACE-Befehl gegeben.
Die Befehlsfolge muss so lange fortgesetzt werden, bis der letzte Datensatz der Primaerdatei bearbeitet wurde.

Sie werden sicherlich bemerkt haben, dass diese Art der Bearbeitung, d. h. fuer jeden Datensatz die gleiche Befehlsfolge ablaufen zu lassen, keinesfalls effektiv ist. Wir wollten Ihnen anhand des vorangehenden Beispiels lediglich die Wirkungsweise der gleichzeitigen Bearbeitung von Dateien demonstrieren, im naechsten Kapitel "Programmieren" werden Sie kennenlernen, wie derartige gleichartige Befehlsfolgen als Programmschleifen ausgebildet werden koennen.
Im Bild 74 ist die eben beschriebene Befehlsfolge noch einmal dargestellt.

```
USE bestand
INDEX ON teilnr TO teilbes
SELECT 1
USE aufpos ALIAS auf
SELECT 2
USE bestand INDEX teilbes ALIAS be
SELECT 1
SET RELATION TO teilnr INTO be
REPLACE betrag WITH be->preis * anz
SKIP
REPLACE betrag WITH be->preis * anz
SKIP
  .
  .
```

Bild 74 Befehlsfolge fuer gleichzeitiges Bearbeiten von Datenbankdateien

Bild 73 zeigte die Datei "aufpos" vor der Bearbeitung, d.h. der "betrag" war bei allen Datensaetzen 0.00. Bild 75 zeigt die nunmehr bearbeitete Datei "aufpos". Das Feld "betrag" enthaelt die Kosten gleicher Teile pro Auftrag.

```
: USE aufpos
: LIST
   Satznr.  AUFNR    TEILNR    ANZ    BETRAG
        1    4567     40359      1    280.00
        2    4567     14043      1      2.45
        3    4567     00106      4      0.12
        4    4567     00031      4      2.24
        5    8423     40018      1   1420.00
```

Bild 75 Datenbankdatei "aufpos" nach der Bearbeitung

Der Befehl

SET RELATION TO

ohne Parameter loest die Verknuepfung von Datenbankdateien, die Dateien bleiben aber geoeffnet.
Fuer den Nutzer wird es natuerlich wichtig sein, nicht nur die Kosten fuer einen Teiltyp pro Auftrag zu erfahren, sondern den Gesamtpreis (d. h. die Kosten aller Teile) pro Auftrag.

Unter der Ueberschrift "Verdichten von Dateien" lernten Sie im Abschnitt 3.5.5. bereits den Befehl TOTAL kennen, mit dem alle Saetze, die in einem spezifizierten Feld den gleichen Schluesselwert besitzen, zusammengefasst, addiert und in eine neue Datei uebernommen werden koennen.

Wir verdichten also die Datei "aufpos", indem wir die Teile, die zur gleichen Auftragsnummer gehoeren, zusammenfassen. Die Datei, in die die so behandelten Datensaetze uebernommen werden, soll "rech" heissen.

Bevor die Datei "aufpos" verdichtet wird, muss sie nach der Auftragsnummer indiziert werden. Es wird eine Indexdatei namens "inaufpos" erstellt.

```
USE aufpos
INDEX ON aufnr TO inaufpos
TOTAL ON aufnr TO rech
```

Sie erhalten von REDABAS-4 die Nachricht:

```
5 Saetze addiert
2 Saetze erzeugt
```

Wir listen die Datei "rech" auf und erhalten folgendes Bild:

```
: USE rech
: LIST
 Satznr. #   AUFNR    TEILNR    ANZ    BETRAG
        1     4567     40359     10    284.81
        2     8423     40018      1   1420.00
```

Bild 76  Datenbankdatei "rech"

Die so entstandene Datei "rech" ist an sich nur von temporaerer Bedeutung, denn die Informationen ueber einen Auftrag bezueglich Auftragsnummer, Kundennummer, Betrag, Datum und der Aussage, ob die Rechnung bezahlt wurde oder nicht, sind in der Datei "auftrag" gespeichert (vgl. Abschnitt 3.5.) und werden vom Nutzer letztendlich von dort wieder aufgerufen.

Es kommt darauf an, die Betraege aus der Datei "rech" in die Datei "auftrag" zu uebertragen. Dabei ist zu beachten, dass es sich nicht um neue, zusaetzliche Datensaetze handelt, sondern dass Datensaetze mit gleicher Auftragsnummer in der Datei "auftrag" bereits existieren. Diese Uebertragung bedeutet also eine Aktualisierung vorhandener Informationen.

Fuer das Aktualisieren bietet REDABAS-4 den Befehl

UPDATE ON <feld> FROM <alias> REPLACE <feld> WITH <ausdr>,...

Dieser Befehl ermoeglicht das Ersetzen der Daten bestimmter Felder einer Datei durch den Inhalt entsprechender Felder einer anderen Datei. Beide Dateien muessen in unterschiedlichen Arbeitsbereichen geoeffnet sein, sie muessen ein gemeinsames (bei ON spezifiziertes) Feld besitzen, nach dem sie indiziert oder sortiert sind und das der Zuordnung der einzelnen Saetze dient.

Anmerkung: Der Befehl UPDATE bietet ueber das hier Gesagte (und fuer dieses Beispiel Wesentliche) weitere Moeglichkeiten. Deshalb sei an dieser Stelle auf die ausfuehrliche Beschreibung von UPDATE im Abschnitt 9.3.3. verwiesen.

Verfolgen wir wieder unser Beispiel. Die Datei "auftrag" wird nach Auftragsnummern indiziert und zusammen mit der Indexdatei "aufnrau" im Arbeitsbereich 1 eroeffnet. Die Datei "rech" wird ebenfalls nach Auftragsnummern indiziert, und diese beiden Dateien werden im Bereich 2 eroeffnet. Fuer die Ausfuehrung des UPDATE-Befehls wird die Datei "auftrag", d. h. Arbeitsbereich 1 aktiviert.

```
SELECT 1
USE auftrag ALIAS nr
INDEX ON aufnr TO aufnrau
SELECT 2
USE rech ALIAS re
INDEX ON aufnr TO rechnr
SELECT 1
UPDATE ON aufnr FROM re REPLACE betrag WITH
betrag+re->betrag, datum WITH DATE()
```

REDABAS-4 meldet nach Ausfuehrung des UPDATE-Befehls:

```
2 Saetze aktualisiert
```

Bevor wir die aktualisierte Datei "auftrag" auflisten, einige Bemerkungen zum obigen UPDATE-Befehl.

Zunaechst zur auesseren Form:
Bei der Eingabe des obigen UPDATE-Befehls werden Sie bemerkt haben, dass die Laenge einer Bildschirmzeile nicht ausréicht. Dies ist im interaktiven Modus (in dem wir bisher ausschliesslich arbeiteten) ohne weiteres moeglich, solange die Maximallaenge eines Befehls von 254 Zeichen nicht ueberschritten wird. Bei Eingabe eines Befehls im interaktiven Modus rueckt die Zeile automatisch ueber den Bildschirmrand hinaus, wenn die Bildschirmbreite nicht ausreicht.

Nun zum Inhalt des Befehls:
Die Datei "auftrag" ist im ersten Arbeitsbereich aktiviert, mit FROM wird ueber den Aliasnamen die Datei "rechnung" angesprochen, aus der der neue Betrag uebernommen wird. Das bei ON spezifizierte Datenfeld ist das beiden Dateien gemeinsame Schluesselfeld. Die Formulierung des Ausdrucks "betrag+re-> betrag" hinter REPLACE ermoeglicht, dass sich der neue Inhalt des Feldes "betrag" aus einer Summe des alten Betrags (Datei "auftrag") und des neuen Betrags (Datei "rechnung") zusammensetzt. Das anschliessende Komma schliesst diese Ersetzung ab und ermoeglicht weitere Ersetzungen.
Wir moechten noch das aktuelle Datum fuer die zu aktualisierenden Datensaetze aufnehmen. REDABAS-4 bietet mit der sogenannten Datum-Funktion

```
DATE()
```

die Moeglichkeit, das aktuelle Datum aus der rechnerintern gespeicherten Datumsangabe zu uebernehmen.

Das Feld "datum" der Auftragsdatei wird entsprechend der Formulierung im UPDATE-Befehl bei den zu ersetzenden Datensaetzen mit dem aktuellen Datum gefuellt. (Im Beispiel 08.10.89)
Bild 77 zeigt die aktualisierte, nach Auftragsnummern indizierte Datei "auftrag".

| Satznr. | AUFNR | KUNR | BETRAG | DATUM | BEZA |
|---|---|---|---|---|---|
| 1 | 0123 | 421 | 15.00 | 23.01.89 | .T. |
| 5 | 0211 | 789 | 38.18 | 23.02.89 | .T. |
| 2 | 0310 | 910 | 38.38 | 02.02.89 | .F. |
| 3 | 0534 | 678 | 293.85 | 15.02.89 | .T. |
| 4 | 1230 | 910 | 34.64 | 15.02.89 | .T. |
| 10 | 1234 | 850 | 35.54 | 27.02.89 | .T. |
| 6 | 2442 | 123 | 284.81 | 27.02.89 | .F. |
| 7 | 2676 | 048 | 0.00 | . . | .F. |
| 9 | 4567 | 789 | 284.81 | 08.10.89 | .F. |
| 11 | 6677 | 750 | 1420.00 | 17.02.89 | .F. |
| 8 | 8423 | 421 | 1420.00 | 08.10.89 | .F. |

Bild 77 Aktualisierte, nach Auftragsnummern indizierte Auftragsdatei namens "auftrag"

### 3.11.3. Verknuepfen von zwei Datenbankdateien

Fuer das Verknuepfen von zwei Dateien zu einer dritten bietet REDABAS-4 einen sehr leistungsfaehigen Befehl, der in seiner allgemeinen Form lautet:

JOIN WITH<alias> TO <db-datei> FOR<bedingung> FIELDS<feldfolge>

Mit diesem Befehl werden Daten aus zwei in unterschiedlichen Arbeitsbereichen geoeffneten Datenbankdateien in einer neuen Datenbankdatei zusammengefasst.
Zunaechst muessen die beiden Ursprungsdateien in zwei Arbeitsbereichen eroeffnet werden. Die verknuepfte, neu entstehende Datei (bei TO spezifiziert) enthaelt Daten aus der im aktuellen Arbeitsbereich eroeffneten Datei und aus der im JOIN-Befehl nach WITH spezifizierten Datei. Hinter FIELDS sind in der <feldfolge> die zu uebernehmenden Felder anzugeben, d.h. dort wird die Struktur der neuen Datei bestimmt. Ohne Angabe bei FIELDS werden alle Felder aus den beiden Ursprungsdateien uebernommen.

In der FOR-Klausel werden Bedingungen spezifiziert. Hier wird entschieden, welche Saetze der ersten Datei mit welchen Saetzen der zweiten Datei zu neuen Datensaetzen der dritten Datei kombiniert werden. Die Formulierung der FOR-Klausel hat mit groesster Sorgfalt zu erfolgen. Wurde sie ungeeignet oder zu grosszuegig gehandhabt, kommt es zu unueberschaubaren Dateigroessen. Wurden z. B. mit JOIN zwei Dateien mit je 1000 Saetzen bei stets erfuellter FOR-Bedingung vereinigt, muesste die dritte Datei 1 000 000 Datensaetze aufnehmen!

Wie ist der interne Ablauf beim Verknuepfen von zwei Dateien?
Am Anfang weist der Datensatzzeiger in beiden Dateien auf den ersten Datensatz. Nun wird der erste Satz der ersten Datei gemaess der FOR-Bedingung nacheinander mit jedem Satz der zweiten Datei verglichen. Jeder Satz, der dieser Bedingung genuegt, wird in die neue Datei geschrieben.
Ist die Bedingung nicht erfuellt, entsteht kein Datensatz. Sind alle Saetze der zweiten Datei verglichen, wird derselbe Vorgang fuer den zweiten Datensatz wiederholt. Das geht so lange vor sich, bis jeder Datensatz der ersten Datei mit jedem Datensatz der zweiten Datei verglichen wurde.

In einem Beispiel sollen in eine Datei namens "kundrech" die Kunden mit ihrer vollstaendigen Adresse, dem Betrag der noch zu bezahlenden Rechnung und deren Erstellungsdatum aufgenommen werden. Die Informationen sind aus der Datei "auftrag" und der Datei "kunden" zu entnehmen. Es werden aus beiden Dateien jeweils die Saetze vereinigt, die eine gemeinsame Kundennummer haben.

Die entsprechende Befehlsfolge lautet:

```
SELECT 1
USE auftrag ALIAS auf
SELECT 2
USE kunden ALIAS ku
SELECT 1
JOIN WITH ku TO kundrech FOR (kunr=ku->kunr .AND. .NOT. beza
 .AND. betrag>0) FIELDS ku->vorname,ku->name,ku->plz,ku->ort,
ku->strasse,betrag,datum
```

Bedingungen fuer die Schaffung neuer Datensaetze sind neben gleichen Kundennummern, dass schon Rechnungsbetraege vorliegen, diese aber noch nicht bezahlt sind. In diesem Beispiel wurde also die Bildung neuer Datensaetze von mehreren Bedingungen gleichzeitig abhaengig gemacht. Damit wurde die Moeglichkeit demonstriert, den Befehl JOIN selektiv anzuwenden.

In der FIELDS-Klausel wurden die Felder der neuen Datei spezifiziert. Felder der aktivierten Datei (im Arbeitsbereich 1 ist das die Datei "auftrag") muessen nicht gekennzeichnet werden, Felder aus der anderen, bei WITH spezifizierten Datei erhalten den Aliasnamen vorangestellt.

Die verknuepfte Datei "kundrech" enthaelt 5, den bei FOR spezifizierten Bedingungen entsprechende Datensaetze.

Bild 78 zeigt die Datei "kundrech", die mit dem Befehl LIST OFF ausgegeben wird. Der Zusatz "OFF" hat die Funktion, die Ausgabe der vom System intern vergebenen Satznummern zu unterdruecken.

```
: USE kundrech
: LIST OFF
VORNAME NAME       PLZ  ORT       STRASSE          BETRAG DATUM
Peter   Lolle      8122 RADEBEUL  Goethestr. 13     38.38 02.02.89
Gisela  Schmidt    8223 THARANDT  Bahnhofstr. 43   284.81 27.02.89
August  Meier      8122 RADEBEUL  Uferstr. 12     1420.00 08.10.89
Inge    Neumann    8222 RABENAU   Gartenstr. 4     284.81 08.10.89
Karl    Fleischer  8212 FREITAL   Am See 83       1420.00 17.02.89
```

Bild 78 Mittels JOIN gebildete Datenbankdatei "kundrech"

### 3.11.4. Viewgenerator

REDABAS-4 bietet gegenueber seinen Vorgaengersystemen mit dem Viewgenerator ein neues Unterprogramm, das zur Generierung von Verbindungen ( SET RELATION -Befehlen ) zwischen Datenbankdateien dient. Im Abschnitt 3.11.2. lernten Sie das Aufstellen von Verbindungen durch die Wahl von Arbeitsbereichen, das Eroeffnen von Datenbankdateien mit notwendigen Indexdateien und die Formulierung des SET RELATION -Befehls kennen.
Das aufwendige Eingeben dieser Befehle nimmt Ihnen der Viewgenerator ab, indem Sie menuegesteuert Datenbank- mit zugehoerigen Indexdateien auswaehlen und einen Verbindungsausdruck eingeben. Ebenso koennen Sie eine vorher erstellte Maskendatei auswaehlen und eine Filterbedingung definieren. Das Ergebnis wird in einer Viewdatei (Dateityp .VUE) gespeichert und ist jederzeit aufrufbar. Durch eine Viewdatei lassen sich bis zu 9 (mit Katalogdatei) oder 10 (ohne Katalogdatei) Datenbankdateien miteinander in Beziehung setzen, gleichzeitig nach bestimmten

Kriterien ansehen, editieren und ausdrucken. Das An- und Einfuegen (APPEND, INSERT) von Datensaetzen an einzelne Dateien ist damit nicht moeglich, sondern nur fuer die aktivierte Datenbankdatei.

Der Befehl zum Aufruf des Viewgenerators lautet in vereinfachter Form

CREATE VIEW <viewdatei>

Der Name fuer <viewdatei> ist frei waehlbar. Falls keine Dateierweiterung spezifiziert wird, vergibt REDABAS-4 .VUE als Dateityp. Der Viewgenerator arbeitet nach dem Prinzip menuegesteuerter Befehle (vgl. Abschnitt 2.5.). Deshalb sind fuer die Arbeit mit dem Viewgenerator die im Abschnitt 8.4.2. beschriebenen Tasten bzw. Tastenkombinationen gueltig. Insbesondere sind die Unten- bzw. Oben-Pfeiltasten (oder <PgDn> bzw. <PgUp> als Grobeinstellung) zur Fuehrung des Markierungsbalkens auf eine Option und anschliessend <ET> zur Auswahl dieser Option zu betaetigen.

Wir knuepfen an die Aufgabenstellung aus Abschnitt 3.11.2. an und erzeugen mit Hilfe des Viewgenerators eine Verbindung von der Datenbankdatei "aufpos" zur Datenbankdatei "bestand" ueber das gemeinsame und hier gleichbenannte Datenfeld "teilnr". Zusaetzlich soll die Datenbankdatei "auftrag" als nichtverbundene Datei in der Viewdatei aufgenommen werden.
Das Ergebnis unseres Viewgeneratorlaufs soll in der Viewdatei "aufbesv" gespeichert werden.
Schliessen Sie zur Wahrung einer besseren Uebersicht alle Dateien mit dem Befehl

CLOSE ALL bzw. CLOSE DATABASES (mit Ausnahme einer Katalogdatei)!

Geben Sie nun ein:

CREATE VIEW aufbesv

Auf dem Bildschirm erscheint das VIEW-Menue mit dem voreingestellten DBD-Auswahl-Menue (vgl. Bild 79). Im linken Fenster sind alle Datenbankdateien des aktuellen Laufwerks aufgelistet, aus denen diejenigen auszuwaehlen sind, die in die Viewdatei eingehen sollen. Die Reihenfolge der Auswahl bestimmt dabei die Anordnung in den Arbeitsbereichen. Die n-te ausgewaehlte Datei wird im Arbeitsbereich n eroeffnet (1<=n<=10).
Zunaechst ist die Unten-Pfeiltaste so oft zu betaetigen, bis der Markierungsbalken auf AUFPOS.DBD steht. Nun wird diese Datei durch Betaetigen von <ET> als erste ausgewaehlt. Die Datei wird automatisch durch einen Pfeil markiert, und es wird eine Liste aller Indexdateien des aktuellen Laufwerks aufgerufen. Bis zu 7 Indexdateien sind daraus nach demselben Auswahlverfahren wie bei den Datenbankdateien auswaehlbar.

In unserem Fall ist keine besondere Datensatzfolge gefordert. Deshalb beenden wir das Indexdateien-Untermenue mit der Links-Pfeiltaste. Wir waehlen als 2. Datei BESTAND.DBD aus der Liste der Datenbankdateien und TEILBES.IDX aus der Liste der Indexdateien. Wenn beide Dateien ueber ein gemeinsames Datenfeld verknuepft werden sollen, ist es notwendig, dass die 2. Datei nach diesem Datenfeld indiziert ist. TEILBES.IDX ist eine bereits erstellte Indexdatei ueber "teilnr" fuer die Datenbankdatei "bestand".

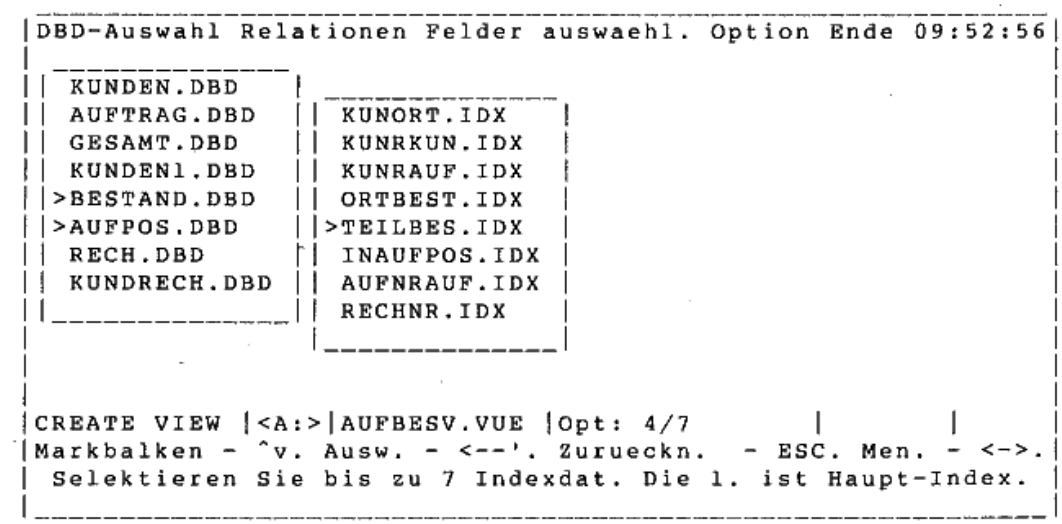



Bild 79 DBD-Auswahl-Menue des Viewgenerators

Bild 79 zeigt, dass TEILBES.IDX ausgewaehlt wurde. Auf eine zu einer Datenbankdatei nicht passende Indexdatei wird in der Anweisungszeile hingewiesen. (Versuchen Sie beispielsweise KUNORT.IDX auszuwaehlen!)
Das Untermenue der Indexdateien wird mit der Links-Pfeiltaste geschlossen. Schliesslich soll als 3. Datenbankdatei AUFTRAG.DBD ausgewaehlt werden, die als unverbundene Datei in die Viewdatei eingehen soll.

Springen Sie aus dem DBD-Auswahl-Menue durch einmaliges Druecken der Rechts-Pfeiltaste in das Relationen-Menue (vgl. Bild 80)! Das Relationen-Menue generiert die Verbindungen zwischen den Datenbankdateien und bestimmt den aktiven Arbeitsbereich. Zunaechst erscheint eine 1. Liste aller im DBD-Auswahl-Menue ausgewaehlten Datenbankdateien (mittleres Fenster in Bild 80). Da der Markierungsbalken auf AUFPOS.DBD voreingestellt ist, brauchen Sie nur <ET> zur Auswahl zu druecken. "aufpos" ist damit als 1. Glied der zu verbindenden Dateien bestimmt. Da "aufpos" im Arbeitsbereich 1 angelegt ist, wird dieser Arbeitsbereich zur Aktivierung vorgesehen.

Nach der Auswahl von "aufpos" erscheint eine 2. Liste der restlichen ausgewaehlten Datenbankdateien (rechtes oberes Fenster in Bild 80). Waehlen Sie die voreingestellte Datei BESTAND.DBD durch <ET> aus! Die Verbindung wird somit von AUFPOS.DBD zu BESTAND.DBD aufgebaut. Auf beiden Dateien ist der Markierungsbalken stehen geblieben. REDABAS-4 verlangt nun die Eingabe eines Verbindungsausdrucks. Das wird deutlich durch einen Pfeil und den Cursor hinter der ausgewaehlten Datei in der 2. Liste.

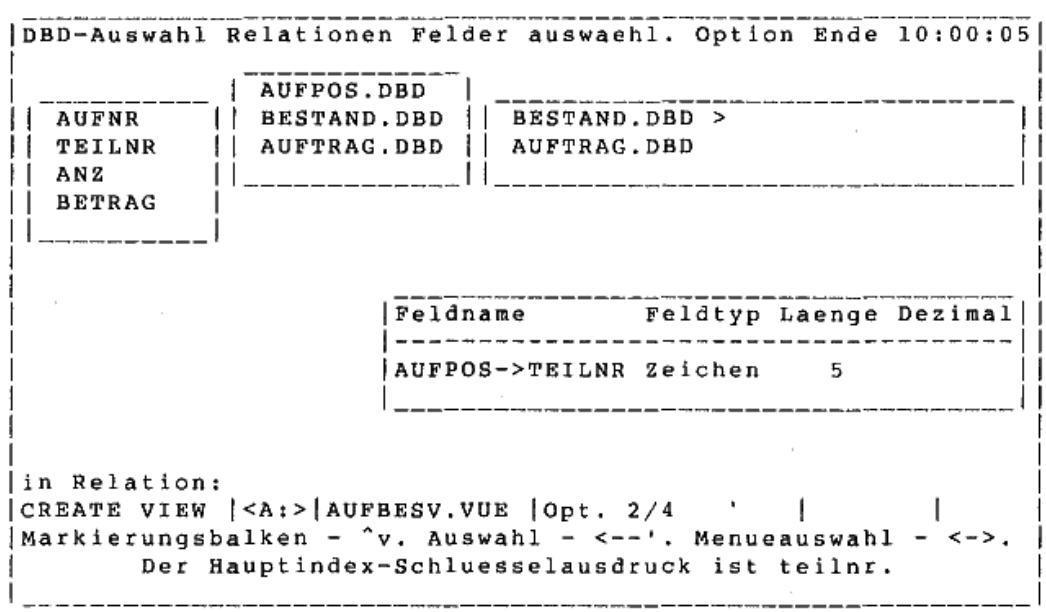



Bild 80 Relationen-Menue des Viewgenerators

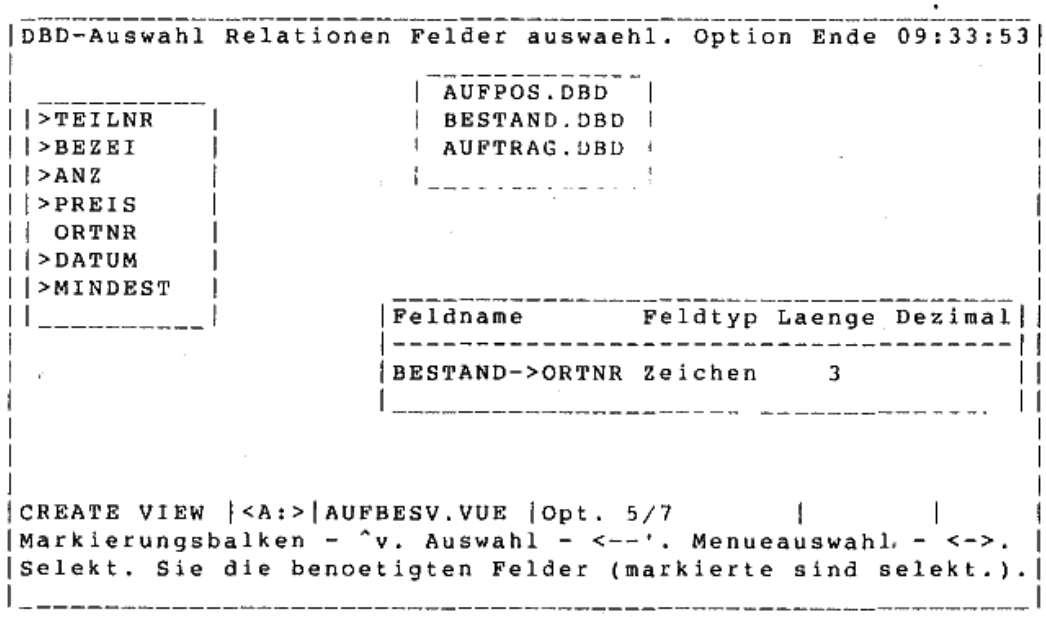



Bild 81 Felder-auswaehlen-Menue des Viewgenerators

Die Meldungszeile weist aus, dass der Hauptindex-Schluessel-ausdruck (von BESTAND.DBD) das Datenfeld "teilnr" ist. Der einzugebende Verbindungsausdruck, gebildet aus Datenfeldern von "aufpos", muss den Datentyp des Schluesselausdrucks ergeben.

In unserem Beispiel soll es das Datenfeld "teilnr" sein, das auch in der Bezeichnung mit dem Datenfeld "teilnr" von "bestand" uebereinstimmt. Durch Betaetigen der <F10>-Taste erhalten Sie eine Liste der Feldnamen von "aufpos" (linkes Fen-

ster in Bild 80) mit zusaetzlicher Darstellung der Feldattribute des eingestellten Feldes (rechtes unteres Fenster in Bild 80).
Bewegen Sie den Markierungbalken mit der Unten-Pfeiltaste auf TEILNR (Bild 80 gibt diesen Zustand wieder) und waehlen Sie dieses Feld mit <ET> aus!

Der Feldname wird als 1. Komponente des Verbindungsausdrucks eingesetzt. Fuer Ihre Aufgabe ist der Verbindungsausdruck komplett und kann mit <ET> abgeschlossen werden. (Sie haetten auch "teilnr" direkt eintippen koennen ohne Hilfe von <F10>.)
Die Aktionszeile zeigt das Ergebnis der Verbindung an:

in Relation: AUFPOS.DBD->BESTAND.DBD

Gehen Sie mit der Links-Pfeiltaste in das Relationen-Menue zurueck!
Betaetigen Sie einmal die Rechts-Pfeiltaste zur Anwahl des Felder-auswaehlen-Menues (vgl. Bild 81)!

Das Felder-auswaehlen-Menue legt fest, welche Felder der in die Viewdatei eingegangenen Datenbankdateien fuer die weitere Arbeit zur Verfuegung stehen. Das Menue haengt mit den Befehlen SET FIELDS ON/OFF und SET FIELDS TO eng zusammen (vgl. Abschnitt 9.3.3.).
War vor Aufruf des Viewgenerators SET FIELDS ON gestellt und wird das Felder-auswaehlen-Menue nicht durchlaufen, wird fuer jeden betroffenen Arbeitsbereich ein Befehl SET FIELDS TO ALL impliziert. Das bedeutet, dass nach einem Aufruf der Viewdatei alle Felder der in die Viewdatei eingegangenen Datenbankdateien ausgewertet werden koennen. Saetze der verbundenen Datenbankdateien werden gemeinsam editiert (EDIT) oder aufgelistet (LIST). War vor Aufruf des Viewgenerators SET FIELDS OFF (Standardwert) gesetzt, bleibt die vorher erklaerte Wirkung aus. Es sind alle Felder ansprechbar, wenn mit dem SELECT -Befehl auf den entsprechenden Arbeitsbereich eingestellt wird. Die Situation aendert sich jedoch, wenn Sie im Felder-auswaehlen-Menue wenigstens ein Feld irgendeiner Datei ausblenden. SET FIELDS wird automatisch auf ON gesetzt und die verbliebenen Felder der verbundenen Dateien sind gleichzeitig ansprechbar.

Das Felder-auswaehlen-Menue ruft eine Liste der im DBD-Auswahl-Menue ausgewaehlten Datenbankdateien auf (mittleres oberes Fenster in Bild 81). Betaetigen Sie einmal die Unten-Pfeiltaste und anschliessend <ET>, um BESTAND.DBD auszuwaehlen! Im linken Fenster wird eine Liste aller zugeordneten Felder angezeigt. Sie sind zu Beginn alle mit einem Pfeil markiert und damit zur Verarbeitung vorgesehen. Im rechten unteren Fenster sind die Feldattribute des aktuellen Feldes ausgewiesen. Bild 81 zeigt den Zustand, dass mit dem ueblichen Auswahlverfahren ORTNR ausgewaehlt wurde. Der Markierungsbalken ist geloescht, und damit ist das Feld von einer spaeteren Verarbeitung im Zustand SET FIELDS ON ausgeschlossen. (Erneutes <ET> wuerde das Feld wieder einbeziehen.) Schliessen Sie auf die gleiche Weise die Felder DATUM und MINBEST aus!

Druecken Sie zweimal die Rechts-Pfeiltaste, um ins Option-Menue zu kommen! Waehlen Sie die Option Filter mit <ET>. Hiermit ist eine Filterbedingung (vgl. Abschnitt 3.2.3.) definierbar, die nur Saetze zur weiteren Verarbeitung zulaesst, die die Filterbedingung erfuellen. Geben Sie beispielsweise folgenden

Ausdruck ein:

BESTAND->BEZEI # "Motor"

und schliessen Sie mit <ET> ab!
Wir verhindern damit die Ausgabe von Saetzen, die im Datenfeld "bezei" den Eintrag "Motor" haben (in unserem Fall Saetze mit "teilnr = 40018"). Die Spezifikation des Feldes mit dem Aliasnamen, der standardmaessig mit dem Datenbankdateinamen uebereinstimmt, ist notwendig, da "bestand" im nichtaktiven Arbeitsbereich 2 angelegt wird. Bei der Eingabe des Ausdrucks merken Sie, wie der eingegebene Text nach links gerollt wird. Da es sich bei der Filterbedingung um keine Querydatei handelt, zaehlt eine gesetzte Bedingung nicht mit in der begrenzten Anzahl (hier 13) gleichzeitig geoeffneter Dateien.
Gehen Sie mit der Unten-Pfeiltaste und <ET> in die Option Maske! Es wird die im Abschnitt 3.10. erstellte Maskendatei "kundensl" als einzige Datei angezeigt. Fuer unsere Viewdatei "aufbesv" muesste erst eine geeignete Maskendatei mit Hilfe des Maskengenerators entworfen werden. Gehen Sie deshalb mit <Esc> zurueck und mit der Rechts-Pfeiltaste zum Ende-Menue! Speichern Sie die Viewdatei "aufbesv" mit <ET>!

Mit dem Speichern der Viewdatei wird diese automatisch aktiviert, und damit werden die in dieser Viewdatei hinterlegten Datenbank- und Indexdateien und gegebenenfalls eine Maskendatei eroeffnet. Die vor Aufruf des Viewgenerators anliegende Arbeitsumgebung wird aufgegeben. Beachten Sie bitte, dass die Anzahl der in eine Viewdatei hinterlegbaren Dateien (einschliesslich einer geoeffneten Katalogdatei) 13 nicht ueberschreiten darf! Erhalten Sie eine diesbezuegliche Fehlermitteilung, dann schliessen Sie eine entbehrliche Datei in einem Arbeitsbereich und rufen den Viewgenerator nochmals auf.

Wenn Sie das Ende-Menue ueber die Option Abbrechen verlassen, wird die urspruengliche Arbeitsumgebung wiederhergestellt und die veranlasste Definition der Viewdatei ignoriert. In unserem Beispiel hatten wir vor Aufruf des Viewgenerators ohnehin alle Dateien geschlossen. Es ist aber moeglich, von einer definierten Arbeitsumgebung auszugehen.

Ueberzeugen Sie sich von der Korrektheit der erstellten Viewdatei "aufbesv", indem Sie die <F6>-Taste betaetigen. Sie erhalten auf dem Bildschirm Bild 82. Beachten Sie weiter, dass FIELDS auf ON gestellt ist. Zur Kontrolle der ausgewaehlten Felder koennen Sie den Befehl LIST STRUCTURE fuer jeden Arbeitsbereich geben. Ausgewaehlte Felder muessen mit einem Pfeil markiert sein.

Geben Sie jetzt den Befehl LIST OFF ein unter der Voraussetzung, dass der selektierte Arbeitsbereich wieder 1 ist! Sie erhalten Bild 83 angezeigt. Die linken Felder "aufnr", "teilnr", "anz" und "betrag" entstammen der Datei "aufpos", die rechten Felder "teilnr", "bezei", "anz" und "preis" kommen aus Datei "bestand". Die Datei "aufpos" (vgl. Bild 75) bestimmt die Datensatzfolge. In den rechten Feldern ist der entsprechende Satz aus "bestand" mit der gleichen "teilnr" ausgewiesen. Die Felder "ortnr", "datum" und "minbest" aus "bestand" sind vereinbarungsgemaess nicht ausgewiesen. Der letzte Satz aus "aufpos" mit "teilnr=40018" wurde wegen der definierten Filterbedingung ausgeschlossen, da der zugehoerige Satz in "bestand" mit " bezei = "Motor" " sie nicht erfuellt.

Es ist moeglich, korrespondierende Saetze der verbundenen Dateien gleichzeitig zu editieren, beispielsweise mit dem EDIT-Befehl. Auf unverbundene Dateien muss dagegen mit dem SELECT-Befehl eingestellt werden. Ueberzeugen Sie sich davon durch Eingabe der Befehle

```
SELECT 3
LIST
```

Es wird die Datenbankdatei "auftrag" angezeigt.

```
: DISPLAY STATUS
Selektierter Arbeitsbereich
Arbeitsber.: 1, Datenbankdatei: A:AUFPOS.DBD  Alias: AUFPOS
               Filter: bestand->bezei # "Motor"
      verknuepft mit: BESTAND
      Verknuepfung  : TEILNR
Arbeitsber.: 2, Datenbankdatei: A:BESTAND.DBD Alias: BESTAND
    Haupt-Indexdatei: A:TEILBES.IDX Schluessel: teilnr
Arbeitsber.: 3, Datenbankdatei: A:AUFTRAG.DBD Alias: AUFTRAG
Datei-Suchpfad:
Aktuelles Laufwerk     : A:
Ziel fuer Druckausgabe : PRN:
Rand =      0
Aktueller Arbeitsbereich =  1
Irgendeine Taste druecken, um fortzusetzen ...
```

Bild 82 DISPLAY STATUS nach Aktivierung der Viewdatei "aufbesv"

```
: LIST OFF
 AUFNR TEILNR ANZ BETRAG TEILNR BEZEI             ANZ  PREIS
 4567  40359    1 280.00 40359  Kurbelwelle        63 280.00
 4567  14043    1   2.45 14043  Ansaugkruemmer    194   2.45
 4567  00106    4   0.12 00106  Federring        1374   0.03
 4567  00031    4   2.24 00031  Sechskantmutter  5972   0.56
```

Bild 83 Auflistung nach Aktivierung der Viewdatei "aufbesv"

Sollten Ihre Kontrollen ergeben, dass die erzeugte Viewdatei nicht Ihren Vorstellungen entspricht, sind jederzeit mit dem Befehl

```
MODIFY VIEW <viewdatei>
```

Aenderungen durchfuehrbar. Im DBD-Auswahl-Menue sind die bisher definierten Datenbankdateien durch einen Pfeil gekennzeichnet. Erneute Auswahl fuehrt zum Ausschluss. Nicht erkennbar sind die frueher ausgewaehlten Indexdateien. Andere vorher eingegebene Werte lassen sich ablesen.
Eine gesicherte Viewdatei kann zu einem spaeteren Zeitpunkt jederzeit mit dem Befehl

```
SET VIEW TO <viewdatei>
```

aufgerufen werden. Damit werden alle geoeffneten Dateien (ausgenommen eine Katalogdatei) aller Arbeitsbereiche geschlossen und die in der Viewdatei hinterlegten Datenbank- und Indexda-

teien mit definierten Verbindungen und gegebenenfalls einer Maskendatei eroeffnet.

## 3.12. Zusammenfassung

Im Verlauf der vorangegangenen Abschnitte haben Sie die wichtigsten Methoden fuer den Aufbau und die Bearbeitung einer Datenbank mit REDABAS-4 kennengelernt. Sie kennen damit alle wesentlichen Befehle, um Dateien zu erstellen, Daten zu laden, zu aendern und zu loeschen sowie Dateien auszuwerten. Sie wissen auch, wie bestehende Dateien bezueglich ihrer Struktur veraendert, sortiert oder indiziert und untereinander verknuepft werden. Sie haben erfahren, wie Sie Ihre Auswertungen, Berichte und Listen mit dem REDABAS-4-Reportgenerator problemlos gestalten und ausgeben koennen und wie Sie mit dem Etikettengenerator arbeiten. Als wesentliche Neuerungen von REDABAS-4 gegenueber seinen Vorgaengersystemen nahmen Sie die Moeglichkeiten des Query-, Masken- und Viewgenerators zur Kenntnis.
Eine weitere wichtige Neuerung von REDABAS-4 ist die Moeglichkeit, sachlich zusammengehoerige Dateien in Katalogdateien zusammenzufassen. Wie Katalogdateien erstellt und genutzt werden, erfahren Sie im Kapitel 6. (Eine detaillierte Beschreibung aller REDABAS-4-Funktionen und -Befehle finden Sie in den Abschnitten 9.2. und 9.3.).

Mit diesen Informationen ausgeruestet, sind Sie in der Lage, Ihr Datenbankproblem auf einem Mikrocomputer zu loesen. Nochmals sei auf die Uebersichtlichkeit und Leistungsfaehigkeit der Loesungswege mit REDABAS-4 als relationalem Datenbankbetriebssystem verwiesen.

REDABAS-4 verfuegt aber ueber diese Moeglichkeiten hinaus noch ueber weitere Befehle, die es Ihnen erlauben, Anwendungsprogramme zu schreiben, ohne dass Sie herkoemmliche Programmiersprachen zu erlernen oder anzuwenden brauchen. So koennen Sie die Befehle, mit denen Sie jetzt vertraut sind, zu Programmen zusammenstellen, diese in Dateien abspeichern und bei Bedarf wieder abrufen. Diese und weitere Moeglichkeiten der Programmierung mit REDABAS-4 lernen Sie im Kapitel 4. dieser Programmtechnischen Beschreibung kennen.

## 4. Programmieren

Wir moechten Sie in diesem Abschnitt mit den Moeglichkeiten von REDABAS-4 bekanntmachen, Befehlsfolgen in Dateien abzuspeichern, wo sie jederzeit abrufbar sind. Sie lernen die entsprechenden Befehle kennen und darueber hinaus Mittel fuer die Bildung von Programmschleifen und Programmverzweigungen.

Damit steht Ihnen im Rahmen von REDABAS-4 eine sehr leistungsfaehige Programmiersprache zur Verfuegung.

Die Wirkungsweise der einzelnen Befehle wird wieder an Beispielen demonstriert, die auf den aus dem Kap. 3. bekannten Dateien "kunden", "bestand", "auftrag" und "aufpos" basieren.

Fuer Ihre jetzige Arbeit mit diesen Dateien wird es notwendig werden, zusaetzliche Daten aufzunehmen und zu bearbeiten. Um aber ebenso jederzeit auf den urspruenglichen Zustand zurueckgreifen zu koennen, empfiehlt es sich, von den Dateien Duplikate anzufertigen. Diese sogenannten "Backup"-Kopien werden unkompliziert auf folgende Weise erstellt:

Wollen Sie beispielsweise eine "Backup"-Kopie der Datei "kunden" erstellen, schreiben Sie:

```
USE kunden
COPY TO kundbak
```

Genauso verfahren Sie mit den anderen Dateien. Der Zusatz "bak" soll auf "Backup"-Kopie verweisen. Indexdateien brauchen nicht kopiert zu werden, sie werden durch Indizieren der Stammdatei je nach Bedarf erzeugt.

### 4.1. Befehlsfolgen

Die abzuspeichernde Befehlsfolge soll in eine Programmdatei geschrieben werden, damit sie - einmal dort abgespeichert - jederzeit unter dem vergebenen Programmnamen wieder abgearbeitet werden kann.

Die Eroeffnung der Programmdatei erfolgt mit

```
MODIFY COMMAND <datei>
```

Der Dateiname (wiederum aus max. 8 Zeichen bestehend und mit einem Buchstaben beginnend) wird intern mit dem Zusatz .PRG durch REDABAS-4 versehen.

Wenn Sie nachfolgend eine Programmdatei namens "progl" mit

```
MODIFY COMMAND progl
```

eroeffnen, wird der gesamte Bildschirm geloescht. In der 1. Zeile erscheint die Ausschrift "Editieren: progl.prg" und danach das Hilfsmenue zur Cursorsteuerung, das mit der <F1>-Taste ab- bzw. angestellt werden kann. Sie befinden sich im REDABAS-4-Texteditor, einem Hilfsprogramm zum Erstellen und

Aendern von Programmen, das einem Textverarbeitungssystem entspricht.
Sie koennen aber Ihr Programm auch mit einem anderen verfuegbaren Texteditor erstellen. (Z.B. wenn Ihr Programmtext mehr als 4096 Bytes einnimmt, denn mit dem REDABAS-4-Texteditor koennen pro Datei nur max. 4096 Bytes Programmtext gespeichert werden). Sie geben in diesem Fall der Textdatei selbst den Programmnamen und muessen auch den Dateityp vorschreiben. In unserem Beispiel: prog1.PRG
Wollen Sie generell fuer Ihre REDABAS-4-Arbeit einen anderen Texteditor vereinbaren, lesen Sie bitte im Abschn. 10.1.3. nach.

Bevor Sie nun eine Befehlsfolge abspeichern, noch einige allgemeine Hinweise fuer das Schreiben des Programms. Beginnen Sie das Programm mit dem Befehl

```
CLEAR
```

Damit wird grundsaetzlich bei Aufruf des Programms als erstes der Bildschirm geloescht und der Cursor rechts vom Bereitschaftszeichen plaziert. CLEAR kann, wie die meisten REDABAS-4-Befehle auch im interaktiven Modus benutzt werden.

Wenn Sie die einzelnen Befehle des Programms eingeben, druecken Sie, wie gewohnt, die <ET>-Taste nach jeder Programmzeile.

Befehlszeilen mit einer Laenge von mehr als 66 Zeichen (maximal 254 Zeichen) koennen durch Beenden der Zeile mit Semikolon und anschliessendes <ET> auf der naechsten Zeile fortgesetzt werden. Andernfalls wird ein Wort, das auf Position 65 noch nicht abgeschlossen ist, auf die naechste Zeile uebernommen. Ein Wort, das auf Position 66 abschliesst, kann mit der Oben-Pfeiltaste und anschliessendem <CTRL-G> auf die vorhergehende Zeile zurueckgeholt werden.

Das Beenden des Programms und die Rueckkehr in die interaktive Ebene von REDABAS-4 realisiert der Befehl

```
RETURN
```

Mit dem Befehl RETURN koennen Sie Ihre abzuspeichernde Befehlsfolge also abschliessen.

Fuer das Eingeben der Befehle nutzen Sie wieder die schon von den Editiermasken bekannten Tastenkombinationen (vgl. Abschn. 3.1.3.).

Als erstes Programm soll nachfolgend das aus dem Abschnitt 3.11.3. bekannte Beispiel der "Kunden mit offenen Rechnungen" als Programmdatei gespeichert werden. Schreiben Sie nach Eingabe von

```
MODIFY COMMAND prog1
```

und der Meldung des Texteditors folgende Befehlsfolge:

```
CLEAR
SELECT 1
USE auftrag ALIAS auf
```

```
SELECT 2
USE kunden ALIAS ku
SELECT 1
JOIN WITH ku TO kundrech FOR (kunr=ku->kunr .AND. .NOT. beza ;
.AND. betrag>0) FIELDS ku->vorname,ku->name,ku->plz,;
ku->ort,ku->strasse,betrag,datum
USE kundrech
? "                          KUNDEN MIT OFFENEN RECHNUNGEN"
?
LIST OFF
RETURN
```

Ein fuer Sie unbekannter Befehl ist der Befehl

```
? <ausdrfolge>
```

Mit diesem Befehl koennen Sie Informationen jeder Art auf dem Bildschirm oder Drucker ausgeben. Wir moechten in der Ueberschrift eine Konstante als Zeichenreihe ausgeben, deshalb muss der Text in Anfuehrungszeichen eingeschlossen sein. Auszugebende Leerzeichen (die Ueberschrift soll ungefaehr mittig plaziert werden) muessen auch innerhalb der Anfuehrungszeichen stehen. Steht das Fragezeichen allein (d. h. ohne <ausdrfolge>), wird die Ausgabe einer Leerzeile veranlasst.

Der Texteditor wird durch Betaetigen der <CTRL-End>- oder <CTRL-W>-Tastenkombination verlassen. Das Programm wird auf Diskette bzw. Festplatte gespeichert.

Anschliessend wird der Bildschirm geloescht, und es erscheint wieder der Doppelpunkt, gefolgt von Leerzeichen und Cursor, als Bereitschaftszeichen fuer die Eingabe von Befehlen.
Jedes Programm wird mit

```
DO <programmdatei>
```

aufgerufen und abgearbeitet. Dieser Befehl kann auch innerhalb eines Programms genutzt werden, um ein sogenanntes Unterprogramm aufzurufen.

Sie koennen das Programm "progl" nun mit dem Befehl

```
DO progl
```

wieder aufrufen und abarbeiten. Dies wird in Bild 84 demonstriert.

```
                    KUNDEN MIT OFFENEN RECHNUNGEN

VORNAME NAME       PLZ  ORT       STRASSE        BETRAG DATUM
Peter   Lolle      8122 RADEBEUL  Goethestr. 13   38.38 02.02.89
Gisela  Schmidt    8223 THARANDT  Bahnhofstr.43  284.81 27.02.89
August  Meier      8122 RADEBEUL  Uferstr. 12   1420.00 08.10.89
Inge    Neumann    8222 RABENAU   Gartenstr. 4   284.81 08.10.89
Karl    Fleischer  8212 FREITAL   Am See 83     1420.00 17.02.89
```

Bild 84 Aufruf und Abarbeiten von "progl"

Sie werden bemerkt haben, dass auch Systemnachrichten waehrend der Abarbeitung auf dem Bildschirm erscheinen, z. B.

```
kundrech.dbd bereits vorhanden, Ueberschreiben erwuenscht?
(J/N) Ja
    5 Saetze zusammengefuegt
```

Mit der Information der ersten Zeile warnt Sie REDABAS-4 vor dem Loeschen einer Datei durch Ueberschreiben.
Moechten Sie auf diese Warnungen verzichten, koennen Sie den Befehl

```
SET SAFETY OFF
```

geben. Waehrend eines Programmlaufes stoeren im allgemeinen Meldungen, die REDABAS-4 ausgibt.
Mit

```
SET TALK OFF
```

koennen Sie die Ausgabe dieser Meldungen (ausser Warnungen) unterdruecken.
Denken Sie aber bitte daran, am Ende Ihres Programms die Standardeinstellungen

```
SET SAFETY ON
```

bzw.

```
SET TALK ON
```

wiederherzustellen.

Es ist jederzeit moeglich, "progl" aufzurufen und abzuarbeiten, unabhaengig davon, ob die Dateien "auftrag" oder "kunden" mittlerweile anderweitig bearbeitet wurden.

Will man Befehle veraendern, erfolgt dies mit

```
MODIFY COMMAND progl
```

Das Programm wird in den Arbeitsspeicher geladen und erscheint zur Bearbeitung auf dem Bildschirm. Nach erfolgter Aenderung wird das neue Programm auf einen freien Platz auf der Diskette gespeichert. Die alte Version des Programms (jedoch stets nur die letzte Version) ist als Backup-Datei unter dem Dateinamen "progl.BAK" weiterhin verfuegbar. Damit ist sie fuer REDABAS-4 nicht mehr im unmittelbaren Zugriff, koennte aber bei Bedarf in eine ".PRG"-Datei umgewandelt und damit reaktiviert werden.
(Dazu dient der Befehl RENAME.)

In einem zweiten Beispiel soll ein sogenannter Kundenbeleg ausgegeben werden, der ausgewaehlte Informationen ueber einen Kunden in Form eines Informationsblattes enthaelt. Diese Informationen sollen Name, Vorname, Kundennr., Adresse, Datum des letzten Auftrages und zu zahlender Betrag sein.

Bevor das entsprechende Programm geschrieben wird, noch die folgenden Informationen:

Mit dem Befehl

@ <koordinaten>

wird es Ihnen ermoeglicht, Ihre Daten auf dem Bildschirm oder auf der Druckliste exakt zu positionieren. Unter "koordinaten" sind konkret Zeile und Spalte zu verstehen, bei der eine zu schreibende Information beginnen soll. Zeile und Spalte sind also Zahlenwerte, die Sie entsprechend Ihrer Bildschirmgroesse oder Ihrer Formularbreite (fuer das Drucken) angeben, d. h. Sie koennen jede moegliche Zeichenposition ansprechen. Die Ausgabe eines Zeichens erfolgt unmittelbar hinter bzw. unter der angegebenen Zahl. Mit

@ 0,0

begaenne also Ihre Information in der 1. Spalte der 1. Zeile. (Das heisst, Spalten und Zeilen werden mit 0 beginnend numeriert.)

Sie koennen den Befehl @ (auf manchen Tastaturen ist statt des @ (Klammeraffe) das Paragraph-Zeichen zu finden und zu benutzen) noch erweitern fuer die Ausgabe von Text oder Daten. Er lautet dann

@ <koordinaten> SAY <ausdr>

Unter <ausdr> werden Text, Namen von Datenfeldern oder Speichervariablen oder entsprechende Verknuepfungen verstanden.

Der Befehl @ kann gut fuer das Formatieren von Druckformularen genutzt werden. Zum einen, weil man Druckpositionen sehr praezis festlegen kann. (Dies ist z.B. wichtig fuer die Ausgabe von Informationen auf vorgedruckte Formulare.)
Zum anderen deshalb, weil man mit diesem Befehl Informationen fuer den Druck markieren kann. Das heisst man kann gezielt nur die Informationen ausdrucken, die mit "@" erzeugt werden und andere Informationen vom Druck ausklammern. Am Anfang des Programms wird deshalb der Befehl

SET DEVICE TO PRINTER

gegeben, der bedeutet: Ausgabe der formatierten Zeilen auf den Drucker, d.h. diese Zeilen werden nur auf dem Drucker und nicht auf dem Bildschirm ausgegeben.

Soll die Ausgabe der Informationen wieder auf den Bildschirm gelegt werden, gibt man ein

SET DEVICE TO SCREEN

Auf diese Weise kann man unkompliziert Formulare, z.B. fuer Rechnungen, erstellen.
Um ein Programm auch waehrend des Programmablaufs noch steuern zu koennen, bietet REDABAS-4 die Moeglichkeit, ueber geeignete Befehle entsprechende Daten einzugeben. Gibt man einen derartigen Befehl an, haelt das Programm an und wartet auf eine Eingabe ueber die Tastatur. Ist diese Eingabe erfolgt, wird die Verarbeitung automatisch fortgesetzt. Einer dieser Befehle lautet

```
ACCEPT <ausdrC> TO <speichvarC>
```

<ausdrC> ist eine wahlfreie beliebige alphanumerische Information, die auf dem Bildschirm erscheint und als Erlaeuterung fuer die nachfolgende Eingabe gedacht ist, die in <speichvarC> geschrieben werden soll. I.a. wird bei <ausdrC> ein Nachrichtentext direkt angegeben, d.h. als Zeichenreihenkonstante (also in Zeichenreihenbegrenzer eingeschlossen) formuliert.

Doch nun zurueck zu unserem zweiten Beispiel, dem gewuenschten Ausdruck eines Kundenbeleges, wie ihn Bild 85 zeigt.

```
┌──────────────────────────────────────────────────┐
│                                                  │
│                  Kundenbeleg                     │
│                                                  │
│   Weber, Franz                                   │
│                                                  │
│   Kundennummer: 850          Heideweg 66         │
│                              THARANDT            │
│                              8223                │
│                                                  │
│   letzter Auftrag vom: 27.02.89                  │
│                                                  │
│   Betrag:    35,54 M                             │
└──────────────────────────────────────────────────┘
```

Bild 85 Zusammenstellung wesentlicher Informationen ueber einen Kunden

Das Programm soll "prog2" heissen, und Sie schreiben

```
MODIFY COMMAND prog2
```

Sie befinden sich jetzt wieder im REDABAS-4-Texteditor. Sie koennen in Ihr Programm auch Erlaueterungen einfuegen. Dazu dient der Befehl

```
* <text>
```

Soll eine Befehlszeile kommentiert werden, so kann dieser Kommentartext mit && eingeleitet werden. Steht am Anfang der Zeile ein "*" oder am Ende des Befehls &&, so wird der folgende Text von REDABAS-4 als Kommentar verstanden und bei der Programmausfuehrung uebersprungen. In unserem Beispiel wollen wir zur Kenntlichmachung folgenden Kommentar einfuegen:

```
* Programm Kundenbeleg
```

und die naechsten zwei Befehle durch && kommentieren. Die Ausgabe der Systemnachrichten wird unterdrueckt mit

```
SET TALK OFF    && Unterdruecken der Meldungen
```

Das eigentliche Programm beginnt wieder mit der Bildschirmloeschung. Danach wird die Ausgabe dem Drucker zugewiesen und die Ueberschrift auf die 2. Zeile in die 23. Spalte positioniert:

```
CLEAR      && Loeschen Bildschirm
SET DEVICE TO PRINTER
@ 1,22 SAY "Kundenbeleg"
```

Die Informationen, die wir fuer den Kundenbeleg brauchen, muessen zwei unterschiedlichen Dateien entnommen werden. In der Datei "kunden" sprechen wir die Datenfelder "name", "vorname", "plz", "ort" und "strasse" an, in der Datei "auftrag" die Felder "betrag" und "datum". Die Kundennummer ist im Feld "kunr" in beiden Dateien vorhanden und stellt die logische Verbindung zwischen ihnen her. Im folgenden wird die Datei "kunden" im ersten Arbeitsbereich und die zuvor nach Kundennummern ueber die Indexdatei "kunrauf" geordnete Datei "auftrag" im zweiten Arbeitsbereich eroeffnet. (Diese Indexdatei wurde zur Erhoehung der Verarbeitungsgeschwindigkeit eingefuehrt.)

Wir eroeffnen zuerst den zweiten Arbeitsbereich und erst anschliessend den ersten, da wir im ersten Bereich weiterarbeiten wollen, und verbinden beide Dateien.

```
SELECT 2
USE auftrag INDEX kunrauf ALIAS auf
SELECT 1
USE kunden ALIAS ku
SET RELATION TO kunr INTO auf
```

Der Name des Kunden ist bekannt. Um die Anschrift zu finden, muss die Datei "kunden" durchsucht werden.

Die Eingabe des Namens und Vornamens wird auf dem Bildschirm durch eine entsprechende Aufforderung eingeleitet.

Der Name wird in die Speichervariable "Sname" und der Vorname in die Speichervariable "Svorname" gebracht und durch den LOCATE-Befehl mit den entsprechenden Feldern der Datei "kunden" verglichen. Dabei wird der Teilzeichenreihen-Operator angewendet, um auch unvollstaendig eingegebene Namen aufzufinden.

```
ACCEPT "Bitte Namen eingeben: " TO Sname
ACCEPT "Bitte Vornamen eingeben; " TO Svorname
LOCATE FOR (Sname$name .AND. Svorname$vorname)
```

Stimmen die Namen ueberein, werden aus der Datei "kunden" die erforderlichen Angaben in den Kundenbeleg uebernommen.
Der Befehl

```
@ 2,0 CLEAR
```

loescht von den spezifizierten Koordinaten aus den Bildschirm, d.h. die beiden Eintragungen werden auf Veranlassung dieses Befehls geloescht. Es folgen die @-Befehle zur Ausgabe der Informationen:

```
@ 5,5 SAY TRIM(name)+", "+vorname
@ 7,5 SAY "Kundennummer:"+" "+kunr
@ 7,30 SAY strasse
@ 8,30 SAY ort
@ 9,30 SAY plz
@ 12,5 SAY "letzter Auftrag vom:"+" "+DTOC(auf->datum)
@ 14,5 SAY "Betrag:"+STR(auf->betrag,8,2)+ " "+"M"
```

Einige @-Befehle weisen Besonderheiten auf, die wir nachfolgend erlauetern moechten.

Sie lernen hier die Funktion

```
TRIM (<ausdrC>)
```

kennen, die Leerstellen am Ende einer Zeichenreihe entfernt, falls diese vorhanden sind.
In unserem Falle heisst das, der Name wird auf seine tatsaechliche Laenge verkuerzt, und der Vorname kann ohne unnoetige Leerstellen angeschlossen werden.

Weiterhin sehen Sie, dass man innerhalb eines SAY-Befehls mehrere Datenfelder oder Zeichen bzw. Texte mittels "+"-Zeichen miteinander verknuepfen kann, vorausgesetzt, alle Datenfelder einer Befehlszeile sind vom Datentyp "Zeichen".
Will man Felder, die nicht vom Datentyp "Zeichen" sind, in einer Befehlszeile mit Zeichenfeldern ausgeben, muessen diese entsprechend umgewandelt werden.

So bietet REDABAS-4 die Funktion

```
STR(<ausdrN1>[,<ausdrN2>[,<ausdrN3>]])
```

Mit dieser Funktion wird ein numerischer Ausdruck in eine Zeichenreihe umgewandelt. Eine genaue Beschreibung der STR-Funktion finden Sie im Abschnitt 9.2.
Wir moechten uns an dieser Stelle nur auf das Wesentliche beschraenken.
Das 1. Argument stellt den umzuwandelnden numerischen Ausdruck dar, das 2. Argument die Laenge der Zeichenreihe, das 3. Argument die Stellen nach dem Dezimalpunkt.
Es ist zu beachten, dass ein mit der STR-Funktion umgewandelter numerischer Ausdruck die typischen numerischen Eigenschaften verliert und nicht mehr in arithmetischen Operationen weiter verwendet werden kann.

Wir benutzen die STR-Funktion, um den Wert des Betrages in einer Zeile mit dem erlaueternden Text zu schreiben.

Eine weitere Funktion zum Umwandeln eines Ausdruckes ist die Funktion

```
DTOC(<ausdrD>)
```

Es wird ein Datum (Datentyp D) in eine Zeichenreihe umgewandelt. Damit ist gewaehrleistet, dass auch hier der zugehoerige Text mit dem Datum in einer SAY-Klausel stehen kann.

Anschliessend werden mit dem Befehl

```
CLEAR ALL
```

alle Dateien geschlossen und alle Speichervariablen geloescht.

```
SET DEVICE TO SCREEN
```

legt die Ausgabe der formatierten Daten wieder auf den Bildschirm zurueck und muss in jedem Fall gesetzt werden. Sonst

erschienen eventuell in einem spaeteren Programm wichtige Informationen auf dem Drucker anstatt auf dem Bildschirm. Fuer REDABAS-4 gilt allgemein, dass saemtliche Befehle auch ueber das Ende eines Programms hinaus gelten und zwar so lange, bis sie durch Befehle, die deren Wirkung aufheben, ersetzt werden oder bis REDABAS-4 beendet wird. Das gilt ebenfalls fuer das Ansprechen von Dateien. Eine Datei bleibt auch ueber das Ende eines Programms hinaus so lange eroeffnet, bis sie mit dem Eroeffnen einer neuen Datei im gleichen Arbeitsbereich automatisch geschlossen wird.

In unserem Beispielprogramm wird die Ausgabe der Systemnachrichten wieder zugeschaltet und mit RETURN in die interaktive Ebene von REDABAS-4 zurueckgekehrt. Damit sind alle Befehle fuer "prog2" eingegeben, und mit <CTRL-W> oder <CTRL-End> wird die Erstellung des Programms abgeschlossen.

Bild 86 zeigt noch einmal das Programm. Es sei hier darauf verwiesen, dass diese Version des Ausdrucks von Kundenbelegen sehr vereinfacht ist und dass beispielsweise bei falscher Eingabe des Namens das Ergebnis nicht korrekt ist (vgl. hierzu das Programm "progkun2", Bild 89).

```
* Programm Kundenbeleg
SET TALK OFF      && Unterdruecken der Meldungen
CLEAR             && Loeschen Bildschirm
SET DEVICE TO PRINTER
@ 1,22 SAY "Kundenbeleg"
SELECT 2
USE auftrag INDEX kunrauf ALIAS auf
SELECT 1
USE kunden ALIAS ku
SET RELATION TO kunr INTO auf
ACCEPT "Bitte Namen eingeben: " TO Sname
ACCEPT "Bitte Vornamen eingeben: " TO Svorname
LOCATE FOR (Sname$name .AND. Svorname$vorname)
@ 2,0 CLEAR
@ 5,5 SAY TRIM(name)+", "+vorname
@ 7,5 SAY "Kundennummer:"+" "+kunr
@ 7,30 SAY strasse
@ 8,30 SAY ort
@ 9,30 SAY plz
@ 12,5 SAY "letzter Auftrag vom:"+" "+DTOC(auf->datum)
@ 14,5 SAY "Betrag:"+STR(auf->betrag,8,2)+" "+"M"
CLEAR ALL
SET DEVICE TO SCREEN
SET TALK ON
RETURN
```

Bild 86 "prog2" - Programm zur Erstellung eines Kundenbelegs

Wollen Sie nun Kundenbelege ausdrucken, rufen Sie das Programm mit

```
DO prog2
```

auf. Sie werden um die Eingabe des Namens gebeten und erhalten nach kurzer Zeit den entsprechenden Beleg ausgedruckt.

## 4.2. Programmschleifen

Im Abschnitt 3.11.2. lernten Sie im Rahmen des gleichzeitigen Bearbeitens von Dateien eine Befehlsfolge kennen, mit der die Kosten der Teile eines Auftrages und dessen Gesamtkosten ermittelt wurden (vgl. Bild 74). Schon dort wurde erkannt, dass die angewandte Methode, fuer jeden Datensatz die gleiche Befehlsfolge ablaufen zu lassen, recht uneffektiv ist.

Fuer derartiges wiederholtes Abarbeiten bietet sich der Einbau einer Programmschleife an, wodurch eine abgegrenzte Befehlsfolge beliebig oft durchlaufen werden kann.

Nachfolgend soll im Programm "rechnung" demonstriert werden, wie durch den Einbau einer Programmschleife das mehrfache Eingeben gleicher Befehlsfolgen eingespart wird. Schreiben Sie

```
MODIFY COMMAND rechnung
```

und geben Sie nach der Meldung des Texteditors die im Bild 87 dargestellte Befehlsfolge ein.

```
* Kundenrechnung erstellen
CLEAR
SELECT 2
USE bestand ALIAS be
INDEX ON teilnr TO teilbes
SELECT 1
USE aufpos ALIAS auf
INDEX ON aufnr TO inaufpos
SET RELATION TO teilnr INTO be
DO WHILE .NOT. EOF()
   REPLACE betrag WITH be->preis*anz
   SKIP
ENDDO
TOTAL ON aufnr TO rech
SELECT 3
USE auftrag INDEX aufnrau ALIAS nr
SELECT 4
USE rech ALIAS re
INDEX ON aufnr TO rechnr
SELECT 3
UPDATE ON aufnr FROM re REPLACE betrag WITH ;
betrag + re->betrag,datum WITH DATE()
CLEAR ALL
RETURN
```

Bild 87 Programm "rechnung", mit Programmschleife realisiert

Wir moechten einige Erlaueterungen zu den noch unbekannten Befehlen geben.

Der Befehl fuer das Eroeffnen einer Programmschleife lautet

```
DO WHILE <bedingung>
```

Die diesem Befehl folgenden Befehle werden solange durchlaufen, wie die <bedingung> erfuellt ist.

Innerhalb der Programmschleife werden die Befehle ausgefuehrt, bis mit

```
ENDDO
```

das Ende der DO WHILE-Schleife markiert ist. Danach wird zum DO WHILE-Befehl zurueckgesprungen und die <bedingung> erneut ausgewertet.
Ist die <bedingung> nicht mehr erfuellt, wird die Schleife verlassen und das Programm mit dem auf ENDDO folgenden Befehl fortgesetzt.

Im vorliegenden Beispiel lautet der Beginn der Programmschleife

```
DO WHILE .NOT. EOF()
```

Das heisst, die Schleife soll solange ausgefuehrt werden, bis das Ende der Datei (hier das der Datei "aufpos") erreicht wird. Ist dies erreicht, wird mit den Befehlen

```
TOTAL ...
SELECT 3 usw.
```

fortgefahren.

Die weiteren Befehle zur Erstellung der Rechnung werden hier nicht naeher erlaeutert, da sie schon aus dem Abschnitt 3.11.2. bekannt sind. (Die weiteren Dateien wurden in den Arbeitsbereichen 3 und 4 eroeffnet.)

Bemerkenswert ist noch, dass es wegen der besseren Uebersichtlichkeit und Lesbarkeit eines Programms zweckmaessig ist, die Befehle, die zur DO WHILE-Schleife gehoeren, etwas einzuruecken.
Anstelle der im Abschnitt 3.11.2. beschriebenen und im interaktiven Modus einzugebenden Befehlsfolgen rufen Sie jetzt mit

```
DO rechnung
```

das Programm zur Erstellung der Kundenrechnung auf.

Sie haben wieder die Moeglichkeit, in Ihrem Programm mit SET SAFETY OFF die Warnungen bzw. mit SET TALK OFF alle Systemnachrichten auszublenden.

Die Anwendung einer Programmschleife soll noch an einem zweiten Beispiel veranschaulicht werden. Ausgangspunkt dafuer ist die Erstellung des Kundenbeleges, wie Sie ihn im vorigen Abschnitt kennenlernten. Wollen Sie eine Anzahl von Kundenbelegen drucken, muessen Sie jedesmal das in Bild 86 dargestellte "prog2" mit "DO prog2" aufrufen. Und bei jedem Aufruf muessen zwei Dateien eroeffnet, d. h. von der Diskette in den Arbeitsspeicher gelesen werden.

Auch hier bietet sich der Einbau einer Programmschleife an, die je nach der Zahl der gewuenschten Kundenbelege abgearbeitet wird.

Fuer die Erstellung des Programms namens "progkunl" moechten wir uns das Eintippen der dem Programm "prog2" sehr aehnlichen Befehlsfolge ersparen. Rufen Sie "progkunl" mit

```
MODIFY COMMAND progkun1
```

auf. Sie befinden sich im REDABAS-4-Texteditor. Druecken Sie die Tastenkombination <CTRL-K> und anschliessend <R>. Das ist ein Befehl des Texteditors zum Einlesen bzw. Einfuegen einer Datei. Wir moechten die Programmdatei "prog2" uebernehmen und geben auf die Aufforderung "Dateiname eingeben:" den vollstaendigen Dateinamen ein:

```
prog2.prg
```

Das Programm "prog2" wird damit nach "progkun1" kopiert. ("prog2" bleibt trotzdem erhalten). Wir brauchen nun nur noch einige Befehle einzufuegen bzw. die Stellung einiger Befehle zu veraendern.
Bild 88 zeigt das aus "prog2" abgeleitete und modifizierte Programm "progkun1".

```
* Programm Kundenbeleg
SET TALK OFF  && Unterdruecken der Meldungen
SELECT 2
USE auftrag INDEX kunrauf ALIAS auf
SELECT 1
USE kunden ALIAS ku
SET RELATION TO kunr INTO auf
weiter ="J"
DO WHILE UPPER(weiter) <> "N"
 CLEAR
 SET DEVICE TO PRINTER
 @ 1,22 SAY "Kundenbeleg"
 ACCEPT "Bitte Namen eingeben: " TO Sname
 ACCEPT "Bitte Vornamen eingeben: " TO Svorname
 LOCATE FOR (Sname$name .AND. Svorname$vorname)
 @ 2,0 CLEAR
 @ 5,5 SAY TRIM(name)+", "+vorname
 @ 7,5 SAY "Kundennummer:"+" "+kunr
 @ 7,30 SAY strasse
 @ 8,30 SAY ort
 @ 9,30 SAY plz
 @ 12,5 SAY "letzter Auftrag vom:"+" "+DTOC(auf->datum)
 @ 14,5 SAY "Betrag:"+STR(auf->betrag,8,2)+" "+"M"
 WAIT "Fuer weitere Kundenbelege <ET>, sonst N druecken! ";
 TO weiter
ENDDO
CLEAR ALL
SET DEVICE TO SCREEN
SET TALK ON
RETURN
```

Bild 88 Programm "progkun1" zur Erstellung von Kundenbelegen mit Programmschleife realisiert

Wieder sollen einige unbekannte Befehle bzw. Funktionen erlaeutert werden.

Das Durchlaufen der DO WHILE-Schleife wird hier vom Inhalt einer Speichervariablen abhaengig gemacht. Um die Schleife

ueberhaupt durchlaufen zu koennen, muss die Bedingung so gesetzt sein, dass sie anfangs auf jeden Fall erfuellt ist.

Es wird vor der ersten Abfrage ein "J" (JA) in den Speicher namens "weiter" gebracht:

```
. weiter ="J"
```

Spaeter kann der Speicherinhalt auf "N" (NEIN) gesetzt werden, die darauffolgende Abfrage wird dann zum Verlassen der Programmschleife fuehren. Mit

```
DO WHILE UPPER(weiter) <> "N"
```

wird die Programmschleife eroeffnet. Das bedeutet, die Schleife soll solange durchlaufen werden, wie der Inhalt der Speichervariablen "weiter" ungleich "N" ist.

Sie erinnern sich sicher an die im Abschnitt 3.2.2. beschriebene Funktion UPPER().

Diese Funktion wandelt alle in der Zeichenreihe (die durch Auswertung des als Argument angegebenen <ausdrC> entsteht) vorkommenden Kleinbuchstaben in Grossbuchstaben um. Damit ist gesichert, dass auch eingegebene Kleinbuchstaben als Grossbuchstaben gewertet werden und im vorliegenden Fall mit Gross-N verglichen werden.

Die nach dem DO WHILE-Befehl folgenden Befehle kennen Sie aus dem vorigen Abschnitt. Neu ist der WAIT-Befehl.

In Abhaengigkeit der Eingabe, auf die "gewartet" wird, entscheidet sich, ob die Programmschleife erneut abzuarbeiten ist oder ob zum Programmende uebergegangen wird.
Der Befehl

```
WAIT <ausdrC> TO <speichvarC>
```

veranlasst eine Pause bei der Abarbeitung eines Programms, um ueber die Tastatur eine Eingabe machen zu koennen. Im vorliegenden Fall wird nach dem WAIT-Befehl bei Eingabe eines "N" oder "n" das Programm zum Ende gefuehrt, da damit die Speichervariable "weiter" den Wert N erhaelt und die obige Bedingung nicht mehr erfuellt ist.

Im Unterschied zu den Befehlen ACCEPT und INPUT, wo ganze Worte eingegeben werden, nimmt WAIT nur eine einzelne Taste als Eingabe an und bietet so Moeglichkeiten fuer schnelle Eingaben. WAIT braucht nicht mit <ET> abgeschlossen zu werden.

WAIT kann auch ohne den Zusatz TO <speichvarC> benutzt werden. WAIT veranlasst dann lediglich eine Pause bei der Abarbeitung eines Programms. Nach einem beliebigen Tastendruck wird fortgesetzt. Will man beispielsweise Zwischenergebnisse eines Programms anzeigen lassen, kann man auf diese Weise eine Unterbrechung des Programms veranlassen.

In unserem vorliegenden Beispiel wird die Ausgabe von Kundenbelegen solange fortgesetzt, wie irgend eine Taste ausser "N" gedrueckt wird. Erst mit "N" wird das Programm verlassen.

Bestimmt haben Sie an den beiden gezeigten Beispielen die Vorteile des Programmierens mittels Programmschleife erkannt.

Sie haben die Moeglichkeit (und in der Praxis besteht dazu meist auch die Notwendigkeit), die erstellten Programme zu protokollieren.

Verwenden Sie dafuer den Befehl

```
TYPE <datei.erw> TO PRINT
```

Der Dateiname muss vollstaendig, d.h. mit Dateityp angegeben werden.
Um die letzten beiden Programme "schwarz auf weiss" zu haben, geben Sie ein:

```
TYPE rechnung.prg TO PRINT      ' bzw.
TYPE progkunl.prg TO PRINT
```

## 4.3. Programmverzweigungen

REDABAS-4 bietet zwei Moeglichkeiten, Verzweigungen in einem Programm zu realisieren.

Die erste Variante stellt die folgende Befehlsgruppe dar:

```
IF <bedingung>
    .
    .
    Befehle fuer erfuellte Bedingung
    .
    .
ELSE
    .
    .
    Befehle fuer nicht erfuellte Bedingung
    .
    .
ENDIF
```

Die zweite Variante lautet:

```
DO CASE
   CASE <bedingung>
          .
          .
          Befehle fuer erfuellte 1. Bedingung
          .
          .
   CASE <bedingung>
          .
          .
          Befehle fuer erfuellte n-te Bedingung
          .
          .
ENDCASE
```

Zunaechst soll die erste Moeglichkeit an einem Programm, das Sie bereits im vorigen Abschnitt (s. Bild 88) kennenlernten, demonstriert werden. Das erweiterte Programm, das jetzt "progkun2" heissen soll, sehen Sie im Bild 89 dargestellt.
(Uebernehmen Sie fuer die Erstellung von "progkun2" wieder - wie im Abschnitt 4.2. vorgestellt - "progkun1" und modifizieren Sie.)

```
* Programm Kundenbeleg
SET TALK OFF    && Unterdruecken der Meldungen
SELECT 2
USE auftrag INDEX kunrauf ALIAS auf
SELECT 1
USE kunden ALIAS ku
SET RELATION TO kunr INTO auf
weiter ="J"
DO WHILE UPPER(weiter) <> "N"
   CLEAR
   SET DEVICE TO PRINTER
   @ 1,22 SAY "Kundenbeleg"
   ACCEPT "Bitte Namen eingeben: " TO Sname
   ACCEPT "Bitte Vornamen eingeben: " TO Svorname
   LOCATE FOR (Sname$name .AND. Svorname$vorname)
   IF EOF()
      SET DEVICE TO SCREEN
      CLEAR
      @ 12,5 SAY "Kunde nicht vorhanden!"
      @ 13,5 SAY "-----------------------"
   ELSE
      @ 2,0 CLEAR
      @ 5,5 SAY TRIM(name)+", "+vorname
      @ 7,5 SAY "Kundennummer:"+" "+kunr
      @ 7,30 SAY strasse
      @ 8,30 SAY ort
      @ 9,30 SAY plz
      @ 12,5 SAY "letzter Auftrag vom:"+" "+DTOC(auf->datum)
      @ 14,5 SAY "Betrag:"+STR(auf->betrag,8,2)+" "+"M"
   ENDIF
   WAIT "Fuer weitere Kundenbelege <ET>, sonst N druecken! " ;
   TO weiter
ENDDO
CLEAR ALL
SET DEVICE TO SCREEN
SET TALK ON
RETURN
```

Bild 89 Programm "progkun2" zur Erstellung von Kundenbelegen mit Programmverzweigung

Vergleichen Sie beide Programme ("progkun1" und "progkun2"), erkennen Sie, dass sie sich bis zum LOCATE-Befehl keineswegs unterscheiden. "progkun2" beruecksichtigt jedoch den Fall, dass der gesuchte Datensatz nicht vorhanden ist.
Ein erfolgloser LOCATE-Befehl setzt die EOF()-Funktion auf wahr (vgl. Abschnitt 9.2., Funktionen). In Abhaengigkeit des logischen Wertes der EOF()-Funktion wird in unserem Beispiel verzweigt. (Wird ein Datensatz mit LOCATE gefunden, liefert EOF() den logischen Wert falsch.)

IF EOF(): Auf dem Bildschirm erscheint die Nachricht "Kunde nicht vorhanden!"

ELSE: Sonst, d.h. die EOF()-Funktion liefert den Wert falsch, wird weiter verfahren wie bereits im Programm "progkunl" realisiert, d. h. der Kundenbeleg wird vollstaendig ausgedruckt.

Damit erkennt man die Wirkungsweise des IF-Befehls: Ist die IF-<bedingung> erfuellt, werden die unmittelbar folgenden Befehle ausgefuehrt, andernfalls kommen die Befehle zur Ausfuehrung, die durch ELSE eingeleitet werden. Enthaelt ein IF-Befehl keinen ELSE-Teil, werden bei Nichterfuellung der IF-Bedingung alle Befehle bis zum naechsten ENDIF-Befehl uebergangen. Ein mit IF eingeleiteter Befehlskomplex muss stets mit dem Befehl ENDIF abgeschlossen sein.

Es ist zweckmaessig, anstelle grosser und komplexer Anwendungsprogramme mehrere kleinere zu entwickeln, diese miteinander zu verknuepfen und nur das jeweils gewuenschte aufzurufen.

Eine vorteilhafte Methode ist die Anwendung der Menuetechnik fuer derartig strukturierte Programme. Auf dem Bildschirm werden in Form einer Uebersicht alle vorhandenen Unterprogramme bzw. deren Funktionen angezeigt. Der Nutzer waehlt ueber die Tastatur das von ihm gewuenschte Programm aus, REDABAS-4 arbeitet es ab und kehrt danach zur Programmuebersicht zurueck, der Nutzer kann erneut auswaehlen.

Im folgenden demonstrieren wir die zweite Verzweigungsvariante mit der DO CASE-Befehlsgruppe anhand eines Menueprogramms, das wiederum auf die Ihnen vertraute Problematik eines Reparaturbetriebes orientiert und die bekannten Dateien auswertet. Folgende Funktionen sollen realisiert werden

- das Erfassen neuer Kundenauftraege
- das Erstellen von Rechnungen
- das Anzeigen des Ersatzteilbedarfs
- das Ausdrucken eines Kundenbeleges.

Bild 90 zeigt die Uebersicht, die dem Nutzer zur Auswahl auf dem Bildschirm angezeigt werden soll. Bild 91 gibt das Menueprogramm wieder.

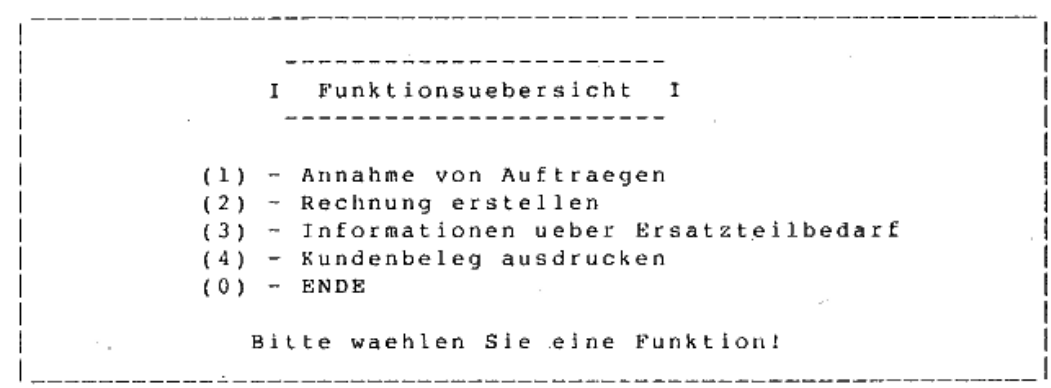



Bild 90 Menueauswahl

```
* Menueprogramm
DO WHILE .T.
   CLEAR
   @ 1,20 SAY "------------------------"
   @ 2,19 SAY "I  Funktionsuebersicht  I"
   @ 3,20 SAY "------------------------"
   @ 5,15 SAY "(1) - Annahme von Auftraegen"
   @ 6,15 SAY "(2) - Rechnung erstellen"
   @ 7,15 SAY "(3) - Informationen ueber Ersatzteilbedarf"
   @ 8,15 SAY "(4) - Kundenbeleg ausdrucken"
   @ 9,15 SAY "(0) - ENDE"
   ?
   WAIT "                  Bitte waehlen Sie eine Funktion!  ";
   TO auswahl
   ?
   DO CASE
      CASE auswahl = "1"
        TEXT
        Unterprogramm "aufnahme" wird noch erstellt
        ENDTEXT
        WAIT
      CASE auswahl = "2"
        DO rechnung
      CASE auswahl = "3"
        USE bestand INDEX ortbest
        REPORT FORM best31 FOR anz<minbest HEADING ;
        "Bedarfsliste"
        ?
        WAIT
      CASE auswahl = "4"
        CLEAR
        @ 5,1 SAY "Bitte den Drucker einschalten!"
        WAIT
        DO progkun2
      CASE auswahl = "0"
        RETURN
    ENDCASE
ENDDO
```

Bild 91  Menueprogramm "menuel"

Nachfolgend werden unbekannte Passagen des Programmes erklaert. Der erste Programmteil realisiert die Anzeige des Menues. Der Befehl

```
DO WHILE .T.
```

ist ein Sonderfall einer Schleifenkonstruktion, da diese Bedingung immer erfuellt ist und nicht zu einer Beendigung einer Programmschleife fuehren kann. (.T. steht fuer den logischen Wert "True", das Gegenteil ist .F. mit der Entsprechung "False". Beide Werte muessen in Punkte eingeschlossen werden, sie stehen jedoch ohne Anfuehrungszeichen.)
DO WHILE .T. erfordert also innerhalb der Schleife einen Ausprungsbefehl.
In unserem Beispiel wird mit den spaeter folgenden Befehlen

```
CASE auswahl = "0"
RETURN
```

die Schleife und auch das Programm "menuel" verlassen.

Nachdem die entsprechenden Ausschriften auf dem Bildschirm positioniert wurden, kann nach dem WAIT-Befehl durch Eingabe einer entsprechenden Ziffer in eines der zur Auswahl stehenden Unterprogramme verzweigt werden. 5 Faelle werden unterschieden.

Die zweite Moeglichkeit der Programmverzweigung, die man auch als Fallauswahl bezeichnen kann, wird mit DO CASE eingeleitet und mit ENDCASE abgeschlossen. Die Anzahl der dazwischenliegenden CASE-Klauseln ist nicht begrenzt. Ist eine der CASE-Bedingungen erfuellt, werden die darauffolgenden uebersprungen.

Im Beispiel wird in Abhaengigkeit von der jeweils gedrueckten Zifferntaste in ein Programm verzweigt:

Ziffer 1:
Es soll das spaeter in Bild 94 dargestellte Programm "aufnahme" abgearbeitet werden, das dem Erfassen neuer Auftraege dient. Dieses sogenannte "Online-Programm" existiert zur Zeit noch nicht (Sie lernen es im Abschnitt 4.5. kennen). Deshalb sollte man in dem Fall keinen Befehl wie z. B. "DO aufnahme" schreiben. Das (nicht vorhandene) Unterprogramm wuerde gesucht, nicht aufgefunden und das gesuchte Menueprogramm wuerde mit einer Fehlernachricht abgebrochen.

Mit den Befehlen

```
TEXT
<text>
ENDTEXT
```

ist es moeglich, beliebige Textzeilen in REDABAS-4-Programmen als Kommentar auszugeben.
Wir geben im Programm den Hinweis, dass das Unterprogramm "aufnahme" noch erstellt wird.

Ziffer 2:
Das aus dem Abschnitt 4.2. und Bild 87 bekannte Programm "rechnung" wird zum Zweck der Erstellung von Rechnungen zur Verarbeitung aufgerufen.

Ziffer 3:
In diesem Fall soll eine Liste ueber den Ersatzteilbedarf ausgegeben werden, wie sie bereits im Abschnitt 3.7. im Bild 57 erstellt wurde.

Ziffer 4:
Es wird das Programm "progkun2" aufgerufen. Zuvor wird eine Aufforderung zum Einschalten des Druckers ausgegeben.

Ziffer 0:
Das Menueprogramm wird beendet, da hier nur der Befehl RETURN zur Ausfuehrung kommt. Das Druecken der Zifferntaste "0" ist die einzige Moeglichkeit, das Menueprogramm zu verlassen.

Das Programm "menuel" kann in dieser Weise abgearbeitet werden. Momentan besteht nur die einzige Einschraenkung bei Druecken von Zifferntaste 1. Druecken Sie diese Taste, erscheint der entsprechend programmierte Hinweis und das Menue

wird wieder angeboten.

Neben dem Aufruf eines Programms mit dem DO-Befehl kann der Programmname auch als Parameter beim REDABAS-Start mitgeteilt werden. Durch Eintippen von

```
REDABAS menuel
```

ruft REDABAS-4 nach Ausgabe der Startmeldung selbstaendig das genannte Programm auf. Beachten Sie aber bitte dabei, dass gegebenenfalls das fuer REDABAS-4-Dateien aktuelle Laufwerk vorangestellt werden muss, also z. B.

```
REDABAS  A:menuel
```

## 4.4. Prozeduren

Bis zum jetzigen Zeitpunkt wurde jedes Programm, das wir erstellten, in einer eigenen Programmdatei abgelegt. REDABAS-4 gestattet jedoch, mehrere Programme in einer Datei, der sogenannten Prozedurdatei zusammenzufassen. So kann eine Prozedurdatei maximal 32 "Unterprogramme" als Prozeduren enthalten.
Wird eine Prozedurdatei zur Verarbeitung aufgerufen, werden saemtliche Programme von der Diskette bzw. Festplatte in den Arbeitsspeicher geladen und bleiben bis zur Beendigung der REDABAS-4-Sitzung (bzw. bis anders lautende Befehle gegeben werden) dort. Damit werden die Zugriffe auf die Diskette oder Festplatte stark verringert, die Verarbeitungsgeschwindigkeit steigt.

Neben der Zeiteinsparung gibt es einen weiteren Vorteil. Obwohl bis zu 32 Prozeduren in einer Prozedurdatei abgelegt werden koennen, zaehlt die gesamte Prozedurdatei als eine einzige Datei. Wenn man bedenkt, dass die Zahl der waehrend der Verarbeitung eroeffneten Dateien auf 15 begrenzt ist, ist leicht erkennbar, dass mit einer Prozedurdatei diese Grenzen erweitert werden koennen.
Am Beispiel unseres Menueprogrammes (vgl. Bild 91) moechten wir Ihnen die Arbeit mit Prozeduren erlauetern. Schreiben Sie

```
MODIFY COMMAND menuepro
```

Nach den ersten beiden Befehlen - einem Kommentar und dem sicherheitshalber durchgefuehrten Schliessen aller Dateien - geben Sie den Befehl

```
SET PROCEDURE TO prozl
```

"prozl" soll also der Name unserer Prozedurdatei sein.
Kopieren Sie anschliessend das Programm "menuel" (im Texteditor <CTRL-K> und <R> druecken und anschliessend den Dateinamen eingeben). Die unter

```
CASE auswahl = "1"
```

aufgefuehrten Befehle werden durch den Prozeduraufruf

```
DO aufnahme
```

ersetzt (wir erstellen diese Prozedur spaeter).

Vor dem "RETURN" wird die Prozedurdatei mit dem Befehl

```
CLOSE PROCEDURE
```

geschlossen. Nach diesen geringfuegigen Aenderungen speichern Sie das (im Bild 92 dargestellte) Programm mit <CTRL-W> bzw. <CTRL-End>.

```
* Programm menuepro
CLEAR ALL
SET PROCEDURE TO proz1
* Menueprogramm
DO WHILE .T.
   CLEAR
   @ 1,20 SAY "-----------------------"
   @ 2,19 SAY "I  Funktionsuebersicht  I"
   @ 3,20 SAY "-----------------------"
   @ 5,15 SAY "(1) - Annahme von Auftraegen"
   @ 6,15 SAY "(2) - Rechnung erstellen"
   @ 7,15 SAY "(3) - Informationen ueber Ersatzteilbedarf"
   @ 8,15 SAY "(4) - Kundenbeleg ausdrucken"
   @ 9,15 SAY "(0) - ENDE"
   ?
   WAIT "                  Bitte waehlen Sie eine Funktion!  ";
   TO auswahl
   ?
   DO CASE
      CASE auswahl = "1"
         DO aufnahme
      CASE auswahl = "2"
         DO rechnung
      CASE auswahl = "3"
         USE bestand INDEX ortbest
         REPORT FORM best31 FOR anz<minbest HEADING ;
         "Bedarfsliste"
         ?
         WAIT
      CASE auswahl = "4"
         CLEAR
         @ 5,1 SAY "Bitte den Drucker einschalten!"
         WAIT
         DO progkun2
      CASE auswahl = "0"
         CLOSE PROCEDURE
         RETURN
    ENDCASE
ENDDO
```

Bild 92  Beispiel eines Prozeduraufrufs

Nun muessen alle in diesem Programm aufgerufenen Unterprogramme bzw. Prozeduren in einer Prozedurdatei zusammengefasst werden. Die Prozedurdatei wird auf bekannte Weise mit

```
MODIFY COMMAND proz1
```

eroeffnet.

Als Prozedur "aufnahme" schreiben Sie nach dem Befehl

```
PROCEDURE aufnahme
```

die kleine Befehlsfolge als Verweis auf die spaetere Erstellung des Programms (vgl. Bild 93).
Holen Sie anschliessend nach dem Befehl

```
PROCEDURE rechnung
```

das Programm "rechnung" (vgl. Bild 87) herein. Gehen Sie dann mit "progkun2" (vgl. Bild 89) genau so vor.
Speichern Sie abschliessend die Prozedurdatei auf bekannte Weise. Bild 93 gibt die Prozedurdatei prozl.prg mit den neuen Befehlen wieder.

```
* Prozedurdatei prozl
PROCEDURE aufnahme
   CLEAR
   ?
   ? "    Dieses Programm wird spaeter erstellt"
   ?
   ?
   WAIT
RETURN

PROCEDURE rechnung
  .
  .
  .
<alle Befehle von "rechnung">
  .
  .
  .
PROCEDURE progkun2
  .
  .
  .
<alle Befehle von "progkun2">
```

Bild 93 Beispiel einer Prozedurdatei

## 4.5. Programmierte Dateneingabe

Zu einem komfortablen Datenbankbetriebssystem gehoeren auch Moeglichkeiten der direkten Erfassung und sofortigen Verarbeitung von Daten in einem laufenden Programm.

Befehle fuer die Erfassung von Daten ueber die Tastatur, die Sie innerhalb dieser Programmtechnischen Beschreibung schon kennenlernten, waren beispielsweise ACCEPT und APPEND. So wird bei Anwendung von APPEND automatisch eine Erfassungsmaske zur Verfuegung gestellt, die aber stets alle Datenfelder in derselben Reihenfolge, wie sie definiert wurden, enthaelt. Oft wird dies fuer eine gezielte Eingabe bzw. Aenderung von einzelnen ausgewaehlten Datenfeldern uneffektiv sein.

Im Abschnitt 4.1. lernten Sie den Befehl @ kennen. @ kann um die sogenannte GET-Klausel erweitert werden und ermoeglicht in dieser Form die direkte Dateneingabe auf folgende Weise:

```
GET <feld>/<speichvar>
```

Diese Klausel bewirkt die Anzeige des Inhalts des spezifizierten Datenfeldes bzw. der spezifizierten Speichervariablen, der vom Nutzer mit dem nachfolgenden Befehl

```
READ
```

ueberschrieben werden kann.

Eine wahlfreie Klausel

```
PICTURE <maske>
```

dient der Formatierung und auch der Plausibilitaetspruefung der Eingabe. Es koennen Daten vom Typ Zeichen, Numerisch, Logisch und Datum eingegeben werden. (Der Cursor kann bei der Eingabe ebenso wie beim gewoehnlichen Editieren vor- und zurueckbewegt werden.)

Im folgenden Beispiel sehen Sie das eben Gesagte demonstriert:

```
PROCEDURE aufnahme
* Erfassen neuer Auftraege
CLEAR
USE auftrag
APPEND BLANK
@ 2,5 SAY "Auftrags-Nr." GET aufnr
@ 3,5 SAY "Kunden-Nr." GET kunr PICTURE "999"
@ 4,5 SAY "Datum" GET datum
READ
RETURN
```

Bild 94 Programm "aufnahme"

Das Programm "aufnahme", ein Unterprogramm des Menueprogramms "menuepro" (s. Bild 92) bzw. eine Prozedur von "prozl" (siehe Bild 93), hat das Erfassen neuer Kundenauftraege zum Inhalt. Diese Auftrage sollen direkt (online) ueber Tastatureingabe mit Auftragsnummer, Kundennummer und aktuellem Datum erfasst werden. (Auftraege sind, wie Sie im Abschnitt 3.5. erfuhren, in der Datei "auftrag" gespeichert.)

Eine entsprechende Erfassungsmaske wird wie folgt erstellt:

Nach dem Loeschen des Bildschirms und dem Eroeffnen der Datei "auftrag" wird mit dem Befehl

```
APPEND BLANK
```

an diese Datei ein leerer Datensatz angehaengt. Der Bildschirm wird nun mit mehreren @-Befehlen gefuellt. Die PICTURE-Klausel gibt in der <maske> eine Zeichenfolge an und definiert damit die gueltige Art der Dateneingabe. 999 bedeutet beispielsweise, dass (in unserem Fall fuer die Kundennummer) nur Ziffern fuer die Eingabe zugelassen sind.

(Natuerlich haetten Sie die Erfassungsmaske auch mit dem in Abschnitt 3.10. beschriebenen Maskengenerator erzeugen koennen und anstelle der @-Befehle im Programm

SET FORMAT TO <maskendatei>

schreiben koennen. Der in der <maskendatei> gespeicherte Befehlsumfang verdoppelt sich, da aus einem obigen @-Befehl ein @ <koordinaten> SAY- und ein @ <koordinaten> GET-Befehl entsteht. Der Einsatz des Maskengenerators lohnt sich jedoch erst bei komplizierterer Maskengestaltung.)

Mit dem Befehl

READ

koennen nun die Daten fuer die drei Felder eingegeben werden.

Mit

RETURN

wird die Befehlsfolge abgeschlossen.

Sie lernten mit dieser Befehlserweiterung eine Moeglichkeit der online-Verarbeitung kennen, naemlich die direkte Erfassung von Daten. Auf diese Weise koennen Sie das Schema fuer die Erfassung und Aktualisierung Ihrer Datenbestaende selbst programmieren und trotzdem den vollen REDABAS-4-Service bzgl. Dateiaktualisierung und Cursorsteuerung nutzen.

Wenn Sie jetzt die Prozedur "menuepro" (siehe Bild 92) aufrufen und in der Menueauswahl die Ziffer 1 waehlen, wird die Prozedur "aufnahme" aktiviert, d.h. Sie koennen in die bereitgestellte Erfassungsmaske neue Auftraege eingeben.

## 4.6. Zusammenfassung

Der Abschnitt 4. zeigte Ihnen insgesamt neue leistungsfaehige Befehle fuer die Programmierung von Anwendungsprogrammen, die sehr komplex sein koennen. Sie erfuhren, wie Befehlsfolgen in Programmdateien zusammengefasst, gespeichert und abgearbeitet werden koennen.

REDABAS-4 ist mit diesen Moeglichkeiten ein sehr leistungsstarkes Datenbankbetriebssystem, das sowohl fuer taegliche Dialogabfragen und -auswertungen als auch fuer sehr komplexe menuegesteuerte Anwendungsprogramme eingesetzt werden kann.

Die fuer die Programmierung wichtigen Befehle wurden in der Reihenfolge beschrieben, wie sie in der Praxis angewandt werden. Demonstrationsbeispiele wurden zum besseren Verstaendnis beigefuegt und ausfuehrlich beschrieben, so dass sie von jedem Anwender leicht nachvollzogen werden koennen, und er sehr schnell erste Erfolge bei der Arbeit mit REDABAS-4 erzielen kann. Einschraenkend muss gesagt werden, dass im Rahmen dieses einfuehrenden Teils der Programmtechnischen Beschreibung nicht alle ausfuehrbaren Befehle und Funktionen und deren Varianten

vorgefuehrt werden konnten. Ziel der bisherigen Beschreibung war es, dem Anwender so schnell wie moeglich einen Einstieg in das Datenbankbetriebssystem REDABAS-4 zu ermoeglichen, ihn mit dessen wesentlichstem Merkmal, der zugehoerigen Programmiersprache vertraut zu machen und damit auch das immense Leistungsspektrum von REDABAS-4 zu verdeutlichen. Im Kapitel 9. finden Sie eine genaue Beschreibung aller REDABAS-4-Funktionen und -Befehle.

Im anschliessenden Kapitel erfahren Sie, wie Sie waehrend einer REDABAS-4-Sitzung Programme aufrufen koennen, die nicht in der REDABAS-4-Sprache geschrieben sind.

# 5. Das Aufrufen externer Programme

## 5.1. Moeglichkeiten des RUN-Befehls

Waehrend REDABAS-4 aktiv ist, lassen sich nicht nur solche Programme abarbeiten, die in der REDABAS-4 Befehlssprache geschrieben sind, sondern auch Kommandos des Betriebssystems DCP und beliebige unter DCP ausfuehrbare Programme. REDABAS-4-Programmdateien rufen Sie mit DO auf, fuer andere Programme benutzen Sie den Befehl RUN.
Waehrend ein mit RUN gerufenes Programm abgearbeitet wird, verbleibt REDABAS-4 im Arbeitsspeicher, und der momentane Arbeitsstand (eroeffnete Dateien, Satzzeiger, Speichervariableninhalte usw.) geht nicht verloren. Sie sparen also viel Zeit und Aufwand dadurch, dass Sie nicht erst REDABAS-4 beenden und nach Ausfuehrung des externen Programms wieder starten muessen. Andererseits muss fuer RUN der Arbeitsspeicher gross genug sein, um sowohl REDABAS-4 als auch das gerufene externe Programm und einen sekundaeren Kommandoprozessor COMMAND.COM des DCP gleichzeitig aufnehmen zu koennen.

Mit RUN koennen Sie folgende Arten von Kommandos und Programmen ausfuehren:
- interne DCP-Kommandos, z.B. DIR, COPY, ERASE, RENAME, MKDIR
- externe DCP-Kommandos und Dienstprogramme, z.B. CHKDSK, COMP, EDLIN
- beliebige Anwendungsprogramme, die unter DCP ausfuehrbar sind, also die Erweiterung .COM oder .EXE im Dateinamen tragen.
- Stapelverarbeitungsprozeduren des DCP, d.h. parametrisierbare Kommandofolgen des DCP, die als Textdatei mit der Erweiterung .BAT vorliegen.

Sie werden feststellen, dass es fuer viele interne Kommandos des DCP auch analoge REDABAS-4-Befehle gibt. Ihr Leistungsumfang ist aber nicht identisch, da DCP als Betriebssystem allgemeinen Belangen dient, waehrend REDABAS-4 der Spezifik einer Datenbankanwendung gerecht wird. Beispielsweise informiert der REDABAS-4-Befehl DIR ueber Datenbankdateien sehr ausfuehrlich und zeigt von anderen Dateien nur die Namen an. Das DCP-Kommando DIR liefert unabhaengig vom Dateityp zu jeder Datei deren Laenge und den letzten Aenderungs- bzw. Erstellungszeitpunkt.
Ausserdem ist in den internen DCP-Kommandos COPY, RENAME, ERASE eine Dateigruppenauswahl ueber Muster moeglich, wo REDABAS-4 nur konkret angegebene Einzeldateien behandelt.

Beispiel: RUN ERASE *.BAK

Auch der Aufruf externer DCP-Kommandos und Dienstprogramme durch RUN kann Vorteile bringen. Z.B. hat jeder Texteditor Vorzuege und Beschraenkungen. Der Zeileneditor EDLIN des DCP ermoeglicht leichtes Transportieren und mehrfaches Kopieren von Zeilengruppen innerhalb einer Textdatei, obwohl er sonst etwas unhandlich ist. Der REDABAS-4-Texteditor konnte aus Platzgruenden nur mit den Hauptfunktionen der Textverarbeitung ausgeruestet werden. Wenn Sie in REDABAS-4 eine Textdatei bearbeiten wollen, koennen Sie aber zwischen dem eingebauten

REDABAS-4-Editor und beliebigen anderen Editoren wie z.B. EDLIN (aufgerufen durch RUN) wechseln. Auf diese Weise benutzen Sie immer den Editor, der Ihre momentanen Belange am besten unterstuetzt.

Beispiel: RUN EDLIN PROG1.PRG

Im Kapitel 10 wird beschrieben, wie Sie anstelle des REDABAS-4-Texteditors von vornherein einen anderen Texteditor in REDABAS-4 einbinden koennen. Dafuer ist der hier erlaeuterte RUN-Befehl nicht geeignet, sondern es muss eine Konfigurierungsdatei erstellt, entsprechend praepariert und vor dem REDABAS-4-Start bereitgestellt werden.

## 5.2. Ausfuehren externer Programme mit CALL

REDABAS-4 bietet zwei Arten des Aufrufs externer Programme:

- RUN fuehrt ein Programm aus, das auch direkt unter Steuerung des Betriebssystems DCP abarbeitbar ist (beschrieben in 5.1.);
- CALL fuehrt ein Programm aus, das als Binaerdatei vorliegt und damit nur als Unterprogramm lauffaehig ist.

Bevor ein Programm ("Modul") beliebig oft mit CALL gestartet werden kann, muss es einmalig durch den LOAD-Befehl von REDABAS-4 in den Arbeitsspeicher geladen werden. Beim Erstellen eines solchen Moduls muessen Sie eine Reihe von Anschlussbedingungen und Einschraenkungen beachten, die im Abschnitt 9.3.3. beim Befehl LOAD beschrieben sind.
Als Programmiersprache bietet sich vorwiegend Assembler an, da dann die Anschlussbedingungen am einfachsten realisierbar sind und die betriebssysteminternen Leistungen direkt angesprochen werden koennen. Das Programm muss compiliert, gebunden und durch das Dienstprogramm EXE2BIN des DCP in eine Binaerdatei verwandelt werden, bevor es von REDABAS-4 aus durch LOAD geladen werden kann.
Sie haben die Moeglichkeit, bei jedem Aufruf des Moduls ueber die WITH-Klausel des CALL-Befehls einen Parameter zu uebergeben und nach der Rueckkehr die Rueckmeldung auszuwerten (siehe Abschnitt 5.3.).
Ob ein Programm durch RUN oder durch CALL abgearbeitet werden soll, muessen Sie bereits bei seiner Erstellung entscheiden und entsprechend beachten.
Der CALL-Aufruf zeichnet sich gegenueber RUN durch wesentlich geringeren Platzbedarf im Arbeitsspeicher und schnellere Abarbeitung aus. Andererseits sind detaillierte Programmierkenntnisse erforderlich, um derartige Moduln zu schreiben. Fuer die Anwendung folgt daraus, dass Sie vor allem dann auf CALL orientieren sollten, wenn zeitkritische oder haeufig angesprochene Arbeitsschritte aufzurufen sind, die geringen Umfang oder ausgepraegte Betriebssystemnaehe aufweisen.
Beispiel zum Aufruf des Moduls "uprol", der als Binaerdatei "uprol.BIN" vorliegt:

```
LOAD uprol              && einmaliges Laden
DO WHILE ...
   ...
  CALL uprol            && wiederholter Aufruf
   ...
ENDDO
RELEASE MODULE uprol    && Freigeben Speicherplatz
```

## 5.3. Parameteruebergabe

Beim Programmaufruf mittels RUN koennen Parameter entweder als konstante Angaben in die Befehlszeile geschrieben werden oder durch die Makroersetzung (siehe Abschnitt 9.4.) flexibel eingebracht werden. Letzteres bedeutet, dass die Parameter - z.B. waehrend eines Programmlauf ermittelt - aus Speichervariablen vom Datentyp "Zeichen" entnommen und an das gerufene Programm uebergeben werden.
Als Beispiel soll der Inhalt einer Speichervariablen "par" in den Aufruf des externen Programms "verarbl" eingesteuert werden. Wenn der Aufruf Bestandteil eines REDABAS-4 - Programms ist, kann der Inhalt der Speichervariablen vorher z.B. durch WAIT/ACCEPT/INPUT vom Bediener ueber Konsole abgefordert werden

```
ACCEPT "Parameter fuer 'verarbl' eingeben: " TO par
RUN    verarbl  &par  ...
```

oder durch eine DO CASE - Mehrweg-Entscheidung ausgewaehlt werden. Sogar den Namen des gerufenen Programms koenen Sie auf diese Weise variabel halten, z.B. nimmt

```
RUN  &komm  &parl  &par2
```

den Programmnamen aus der Speichervariablen "komm", die Parameter aus "parl" und "par2".

Einschraenkend gilt, dass die Parameter bei RUN nur zur Uebergabe in Richtung von REDABAS-4 zum externen Programm genutzt werden koennen. Die Rueckgabe eines Ergebnisses aus dem externen Programm an REDABAS-4 ueber Parameter, d.h. direkt in eine Speichervariable, ist nicht moeglich. Sie laesst sich ueber eine Textdatei (siehe Abschnitt 5.4. und 5.5.) oder auf einem weiteren in Abschnitt 5.5. angedeuteten Umweg abwickeln. Auf jeden Fall ist aber eine Datei erforderlich, da der Inhalt von Speichervariablen durch ein mit RUN gerufenes externes Programm nicht direkt ueberschrieben werden kann.

Beim Aufruf eines Moduls durch CALL kann wahlweise ein (einziger) Parameter angegeben werden, wozu die WITH-Klausel dient. In der Form

```
CALL  <modul>  WITH <ausdrC>
```

dient der Parameter nur der Uebergabe in Richtung von REDABAS-4 zum Modul, waehrend eine Rueckmeldung nicht moeglich ist. Die aus der Ausdrucksbewertung resultierende Zeichenreihe, intern abgeschlossen durch hexadezimal 00, wird im Modul verfuegbar.

Verwenden Sie

```
CALL  <modul>  WITH <speichvarC>
```

erhaelt der Modul direkt die Speichervariable (die Adresse des ersten Zeichens in den Registern DS:BX, das Ende durch hexadezimal 00 begrenzt). Er kann den Inhalt unter Beibehaltung der Laenge der Speichervariablen am Ort ueberschreiben. Dabei muessen Sie mit besonderer Sorgfalt vorgehen, damit Sie den weiteren durch REDABAS-4 belegten Speicherbereich nicht unkontrolliert modifizieren.
Da intern nur eine Uebergabeadresse zur Verfuegung steht, kann in der WITH-Klausel keine Parameterfolge angegeben werden. Sollte der Modul mehrere Parameter benoetigen, muessen Sie diese Informationen zu einer geeignet strukturierten Zeichenreihe verketten und im Modul wieder separieren.

## 5.4. Datenaustausch mit externen Programmen

Das Aufrufen eigener Programme oder beliebiger Anwendungsprogramme aus REDABAS-4 wird Sie erst dann zufriedenstellen, wenn Sie mit solchen Programmen Daten beliebigen Umfangs austauschen koennen. Hierfuer gibt es zwei Moeglichkeiten:
- den Datenaustausch ueber sequentielle Textdateien im Trennzeichenformat oder festen Format.
- der Datenaustausch ueber Spezialformate

Beim Datenaustausch ueber Dateien ist unwichtig, ob die externen Programme aus REDABAS-4 heraus mit RUN (oder CALL) aufgerufen werden oder ob REDABAS-4 und die exterenen Programme vom Betriebssystem aus nacheinander ausgefuehrt werden.

Eine Textdatei (ASCII-Datei) fuer den Datenaustausch wird mit dem REDABAS-4-Befehl (vollstaendige Befehlssyntax s. Absch. 9.3.3.)

```
COPY TO <textdatei> [<bereich>] [FIELDS <feldfolge>]
     [WHILE <bedingung>] [FOR <bedingung>]
     SDF / DELIMITED [WITH BLANK/<begrenzer>]
```

durch Selektion aus einer aktivierten Datenbankdatei erzeugt.

Alle ausgewaehlten Felder werden in einem datentypabhaengigen externen Format dargestellt, das von den Formaten zur Datenanzeige bzw. zur Konstantendarstellung abweicht: logische Werte werden ohne einschliessende Punkte als ein Zeichen "F" oder "T" dargestellt, das Datum wird zuzueglich Jahrhundert sortierfaehig angeordnet, Zeichenfelder werden im Fall von DELIMITED mit Begrenzungszeichen versehen und Merkfelder werden uebergangen. Ein Beispiel zeigt die Struktur einer Datei, die DISPLAY-Darstellung des ersten Satzes und sein Aussehen im Format SDF und im Format DELIMITED, falls keine Feldauswahl erfolgt.

```
Feld   Feldname     Typ           Laenge     Dez
   1   AUFNR        Zeichen          4
   2   KUNR         Zeichen          3
   3   INFO         MERK            10
   4   BETRAG       Numerisch        8        2
   5   DATUM        Datum            8
   6   BEZA         Logisch          1
* Gesamt *                         35

: DISPLAY RECORD 1
 Satznummer   AUFNR KUNR INFO    BETRAG DATUM    BEZA
        1      0123  421  -->M     15.00 23.01.87 .T.

0123421    15.0019870123T          nach SDF

"0123","421",15.00,19870123,T     nach DELIMITED
```

Umgekehrt koennen Sie eine durch ein beliebiges Programm erstellte Textdatei, wenn Sie ein derartiges Format aufweist, in REDABAS-4 wieder in eine Datenbankdatei laden. Dazu dient der Befehl

```
APPEND FROM <textdatei>
            SDF / DELIMITED [WITH BLANK/<begrenzer>]
```

(vollstaendige Befehlssyntax s. Abschn. 9.3.3.). Weil in einer Textdatei keine Feldnamen und Strukturinformationen gefuehrt werden, ist eine passende Struktur der Ziel-Datenbankdatei Voraussetzung, d.h. richtige Feldreihenfolge, vertraegliche Feldtypen und uebereinstimmende (fuer SDF) oder ausreichende (fuer DELIMITED) Feldlaenge.

Da der Austausch von Merkfeldern nicht realisiert ist, darf diese Datenbankdatei keine Merkfelder besitzen! (Lediglich Merkfelder am Ende der Satzstruktur werden toleriert.) Die in unserem Beispiel erzeugten Textdateien koennten an eine Datenbankdatei folgender Struktur angefuegt werden:

```
Feld   Feldname     Typ           Laenge     Dez
   1   AUENR        Zeichen          4
   2   KUNR         Zeichen          3
   3   BETRAG       Numerisch        8        2
   4   DATUM        Datum            8
   5   BEZA         Logisch          1
* Gesamt *                         25
```

Das nachfolgende Beispiel zeigt, wie einem durch RUN aufgerufenen Programm "verarb" die Textdatei templ.TXT vom DELIMITED-Format uebergeben wird. Das Programm erzeugt daraus eine Textdatei temp2.TXT vom SDF-Format, die REDABAS-4 in eine geeignete Datenbankdatei "dbd2" uebernimmt:

```
COPY TO templ DELIMITED
RUN verarb <templ.TXT  >temp2.TXT
USE dbd2
APPEND FROM temp2 SDF
```

An diesem Beispiel sehen Sie weiterhin, dass in dem bei RUN angefuegten DCP-Kommando die Umlenkung der Standardausgabe (>) und -eingabe (<) erlaubt ist. Dasselbe gilt fuer deren Kettung ("pipe"|). Weiterhin muss im DCP-Teil natuerlich die Erwei-

terung TXT des Dateinamens angegeben werden.
Der COPY-Befehl mit SDF oder DELIMITED erzeugt, wie oben gezeigt, die Textdatei direkt aus einer Datenbankdatei. Sie haben aber noch mehr Mittel zur Verfuegung, um Textdateien aufzubauen, wobei Sie mehrere Datenbankdateien kombinieren und die Inhalte der Datenbankfelder in Ausdruecken verarbeiten koennen. Dazu benutzen Sie die Protokolldatei, z.B.

```
SET ALTERNATE TO temp3
...
SET ALTERNATE ON
...                              Ausgabe in temp3.TXT
SET ALTERNATE OFF
...
SET ALTERNATE TO
```

Zwischen SET ALTERNATE ON und SET ALTERNATE OFF wird alle sequentielle Anzeige zusaetzlich in die Protokolldatei geschrieben. Mehrere solche Abschnitte sind moeglich. Das Ausgeben von Daten geschieht meist ueber den Befehl ? , in dem Konstanten, Datenbankfelder, Speichervariable, Funktionen oder Folgen beliebiger komplexer Ausdruecke aus solchen Elementen gestattet sind. Beachten Sie bitte, dass der ?-Befehl seine Operanden ins ASCII-Format verwandelt, wenn sie nicht bereits vom Datentyp Zeichen sind. Dieses "Anzeige-Format" unterscheidet sich etwas vom SDF/DELIMITED-Format, z.B. beim Datum und bei logischen Variablen.
Von praktischer Bedeutung ist, dass die Befehlsfolgen, die RUN enthalten und den Datenaustausch vornehmen, nicht nur auf der REDABAS-4-Befehlsebene abarbeitbar sind, sondern auch in REDABAS-4 Programmdateien eingebaut werden koennen.

An Spezialformaten zum Datenaustausch sind DIF, SYLK und WKS fuer Tabellenkalkulations-Software und das externe Datenbankdatei-Format PFS unterstuetzt. Zum Ausgeben der Daten aus REDABAS-4 in die Formate DIF, SYLK und WKS bzw. zur Eingabe von solchen Quellen dienen spezielle Formen der Befehle COPY TO bzw. APPEND FROM. Bei COPY TO entsteht aus der Datenbankdatei ein zeilenorientierter Tabellenaufbau mit den Feldnamen als Spaltenueberschriften. APPEND FROM setzt eine reihenweise (nicht spaltenweise) Speicherung der Tabelle voraus, wobei die Spaltenueberschriften vorher entfernt werden sollten, weil sonst ein stoerender Kopfsatz mit diesen Ueberschriften in die REDABAS-4 - Datenbankdatei kommt.
Den Datenaustausch ueber das PFS-Format fuehren die Befehle EXPORT und IMPORT aus (Befehlsbeschreibung s. Abschn. 9.3.3.).
Bei der Benutzung eines jeden Spezialformates ist zu beachten, dass der Datenaustausch nur dann sinnvoll ablaeuft, wenn das Datenformat (Datentypen, Feldlaengen, Feldanzahl usw.) sowohl mit REDABAS-4 als auch mit der entsprechenden externen Software vertraeglich ist, also die Grenzen b e i d e r Softwareprodukte nicht verletzt.

## 5.5. Ergaenzende Hinweise

Dieser Abschnitt wendet sich vorwiegend an Fortgeschrittene und kann beim Einarbeiten in REDABAS-4 uebergangen werden.

- REDABAS-4 sucht die unter RUN auszufuehrenden externen Kommandos/Programme oder Stapelverarbeitungsdateien im aktuellen Verzeichnis des aktuellen Laufwerks und bei Bedarf zusaetzlich in den durch das DCP-PATH-Kommando eingefuehrten weiteren Verzeichnissen. Es gelten allein die DCP-Einstellungen, nicht aber Ihre fuer REDABAS-4-Dateien ueber SET PATH oder SET DEFAULT mitgeteilten Zugriffsmoeglichkeiten.

- Was Sie bezueglich des DCP-Kommando-Interpreters COMMAND.COM ueberpruefen muessen, falls RUN nicht arbeitet, ist in Abschnitt 9.3.3. beim RUN-Befehl erlaeutert.

- Nicht alle Aenderungen des DCP-Zustandes, die waehrend RUN durch entsprechende DCP-Kommandos ausgeloest werden koennen, gelten nach Rueckkehr aus RUN weiter. Das betrifft die DCP-Kommandos, die die "Umgebung des Kommando-Interpreters" veraendern (SET, PROMPT, PATH). Diese Kommandos sind deshalb nur in aufgerufenen Stapelverarbeitungsprozeduren ueberhaupt sinnvoll. Mit Verlassen von RUN wird die veraenderte "Umgebung" verworfen, und die urspruengliche, zum Zeitpunkt des REDABAS-4-Aufrufs geltende "Umgebung" wird wieder aktiv (siehe dazu auch GETENV()-Funktion in Abschn. 9.2).

- Die Rueckmeldung aus einem mit RUN ausgefuehrten Programm in eine Speichervariable ist nur ueber den etwas aufwendigen Umweg einer einzeiligen sequentiellen Uebergabe-Datei moeglich, wofuer im folgenden zwei Wege skizziert werden:
  Eine erste Moeglichkeit verlangt, die Struktur einer speziellen Datenbankdatei mit einem einzigen Zeichenfeld <feldC> zu definieren. Das externe Programm gibt seine Rueckmeldung als sequentielle Datei "aus.TXT" aus, oder es benutzt die Standardausgabe, die im RUN-Befehl durch >aus.TXT (siehe DCP-Dokumentation) auf eine solche Datei umgeleitet wird. Mit

```
APPEND FROM aus SDF / DELIMITED [...]
```

  wird der Inhalt von "aus.TXT" in die Datenbankdatei uebernommen.

```
GO BOTTOM
svar = TRIM(<feldC>)
```

  uebertraegt schliesslich die Rueckmeldung in die Speichervariable "svar".
  Bei einer zweiten Moeglichkeit muss das externe Programm eine von REDABAS-4 mit DO abarbeitbare Programmdatei erzeugen, z.B. ueber >aus.PRG im RUN-Befehl. Es genuegt dabei, wenn die gesamte Datei nur aus

```
svar = "<rueckmeldung>"
```

  besteht, wobei <rueckmeldung> fuer die eigentlich zurueckgemeldeten Daten steht.

```
PUBLIC svar
DO aus
```

  uebertraegt die Rueckmeldung schliesslich in die Speichervariable "svar".
  Fuer Programme, die einen geringen Datenumfang rueckmelden, aber zum zeitkritischen Dauergebrauch in einem Projekt bestimmt sind, sollte eine Realisierung als Modul fuer CALL in Erwaegung gezogen werden.

- Beim Aufruf eines Moduls durch CALL koennen nicht nur Speichervariablen vom Datentyp C, sondern von beliebigem Datentyp verwendet werden. Diese Form ist jedoch Sonderfaellen vorbehalten, da die Speichervariable in ihrer internen Form bereitsteht (Typ D wie N als binaere Gleitkommazahl und L als hexadezimal 01 fuer "wahr" und hexadezimal 00 fuer "falsch"). Fuer die normale Anwendung sollte ein Parameter-Wert eines solchen Datentyps mit REDABAS-4 - Mitteln in den Datentyp C konvertiert und als <speichvarC> ausgetauscht werden.

## 6. Die Benutzung von Katalogdateien

Katalogdateien gestatten die Verwaltung sachlich zusammengehoerender Dateien. Sie erleichtern die Auswahl und Uebersichtlichkeit der zu benutzenden Dateien, indem bei bestimmten Befehlen nur die in der ausgewaehlten Katalogdatei eingetragenen Dateien zur Verarbeitung angeboten werden. Insofern wirkt eine Katalogdatei wie ein Filter fuer Dateien.

In eine Katalogdatei koennen Eintragungen von Datenbank-, Report-, Etiketten-, Index-, Masken-, Query-, Screen- und Viewdateien vorgenommen werden. Sie koennen auf einer Diskette oder der Festplatte mehrere Katalogdateien anlegen, z.B. je eine fuer einen bestimmten Nutzerkreis oder ein Sachgebiet. Es ist auch moeglich, eine Datei in mehrere Katalogdateien aufzunehmen, z.B. eine von allen Nutzerkreisen gemeinsam genutzte Datei.

Katalogdateien sind Datenbankdateien mit fest vorgegebener Struktur und dem Dateityp .CAT . Eine andere Erweiterung wird nicht angenommen. Sie koennen wie jede andere Datenbankdatei angezeigt oder editiert werden.

Beim erstmaligen Erzeugen einer Katalogdatei auf einem Datentraeger wird automatisch im aktuellen Verzeichnis des aktuellen Laufwerks eine Hauptkatalogdatei mit dem Namen CATALOG.CAT erstellt. Eine Hauptkatalogdatei ist nach ihrer Struktur ebenfalls eine Katalogdatei, in der aber nur Eintragungen ueber vom Nutzer spezifizierte Katalogdateien gefuehrt werden.
In jedem Verzeichnis bzw. Unterverzeichnis eines Datentraegers kann eine Hauptkatalogdatei angelegt werden. Voraussetzung dafuer ist, dass in einem REDABAS-4-Lauf sich dieser Datentraeger im aktuellen Laufwerk befindet und das jeweilige Verzeichnis als aktuelles Verzeichnis eingestellt ist.

### 6.1. Erstellen, Oeffnen und Schliessen einer Katalogdatei

Mit dem Befehl

```
SET CATALOG TO [<katalogdatei>/?]
```

wird eine Katalogdatei erstellt bzw. geoeffnet oder geschlossen. Wenn Sie bisher ohne Katalogdateien gearbeitet haben, koennen Sie beispielsweise in Ihrem aktuellen Verzeichnis des aktuellen Laufwerks durch Verwendung des Befehls

SET CATALOG TO katrech

erstmalig eine Hauptkatalogdatei CATALOG.CAT und die Katalogdatei "katrech.cat" erzeugen.

Wir empfehlen Ihnen bei der Angabe der Katalogdatei weder Laufwerk noch Pfad zu spezifizieren, damit die Katalogdatei im gleichen Verzeichnis gespeichert wird, in dem sich die Hauptkatalogdatei befindet. Dadurch koennen Sie jede durch den Befehl SET CATALOG TO ? im Menue angezeigte Katalogdatei durch Auswahl eroeffnen.
Andernfalls entsteht die Katalogdatei in dem spezifizierten Verzeichnis und ist ueber SET CATALOG TO ? nicht auswaehlbar.

Im folgenden Text dieses Abschnitts setzen wir voraus, dass Sie die Voreinstellung der Befehle SET SAFETY, SET TITLE und SET TALK auf ON nicht veraendert haben. Andernfalls wuerden die entsprechenden beschriebenen Nachrichten unterdrueckt.

Nach Bestaetigung der Anfrage

Erzeugen einer neuen Katalogdatei? (J/N)

mit J werden Sie mit der Anforderung:

Eingeben eines Titels fuer Datei katrech.cat:

zur Eingabe eines Titels aufgefordert. Damit koennen Sie den Inhalt Ihrer Katalogdatei charakterisieren, z.B. durch "Katalog fuer Abrechnung". Nach Eingabe des Titels erscheint die Ausschrift

Katalogdatei ist leer.

und die Katalogdatei "katrech.cat" wurde erstellt und im Arbeitsbereich 10 eroeffnet.

Bei erstmaliger Anwendung dieses Befehls (bezogen auf das aktuelle Verzeichnis des aktuellen Laufwerks) wird in diesem Verzeichnis eine Hauptkatalogdatei CATALOG.CAT angelegt und in dieser Verweise zur Katalogdatei "katrech.cat" eingetragen.

Erzeugen Sie fuer unser Beispiel auf die gleiche Art die Katalogdatei "katplan" mit dem Titel "Katalog fuer Planung"!

Haben Sie eine oder mehrere Katalogdateien bereits erstellt, so wird durch Angabe der Katalogdatei im SET CATALOG TO - Befehl die jeweilige Katalogdatei im Arbeitsbereich 10 eroeffnet und eine eventuell vorher eroeffnete andere Katalogdatei geschlossen. Dabei wird untersucht, ob die angegebene Katalogdatei in der Hauptkatalogdatei des aktuellen Verzeichnisses des aktuellen Laufwerks eingetragen ist.

Wenn Sie die Katalogdatei ohne Laufwerk und Pfad spezifiziert haben, so wird mit einer bereits existierenden Katalogdatei nichts Auffaelliges passieren. Wenn Sie jedoch zur Eingabe eines Titels aufgefordert werden, dann war Ihre Katalogdatei nicht in der zugehoerigen Hauptkatalogdatei eingetragen.

Ist die eroeffnete Katalogdatei leer, kommt wie nach dem Anlegen einer Katalogdatei die Ausschrift

Katalogdatei ist leer.

Andernfalls werden die in der Katalogdatei enthaltenen Dateinamen ueberprueft, ob sie in den entsprechenden Verzeichnissen vorhanden sind. Existiert ein Dateiname in der Katalogdatei, aber nicht im Verzeichnis, wird er in der Katalogdatei geloescht. Sie erhalten die Nachricht

<datei> nicht gefunden. Aus dem Katalog geloescht.

Eine andere Art der Eroeffnung einer Katalogdatei (bzw. erstmaligen Erstellung der Hauptkatalogdatei) im aktuellen Verzeichnis des aktuellen Laufwerks leitet der Befehl

SET CATALOG TO ?

ein (s. Abschn. 6.5.). Voraussetzung dafuer ist, dass die Hauptkatalogdatei Eintragungen von Katalogdateien enthaelt, d.h. es darf keine Nachricht

Hauptkatalog ist leer.

erscheinen. Wie Bild 95 zeigt, werden in einem Doppelrahmen die im Hauptkatalog eingetragenen Katalogdateien in Form eines Menues und in einem Einfachrahmen der Titel der voreingestellten Katalogdatei als Kommentar angezeigt.

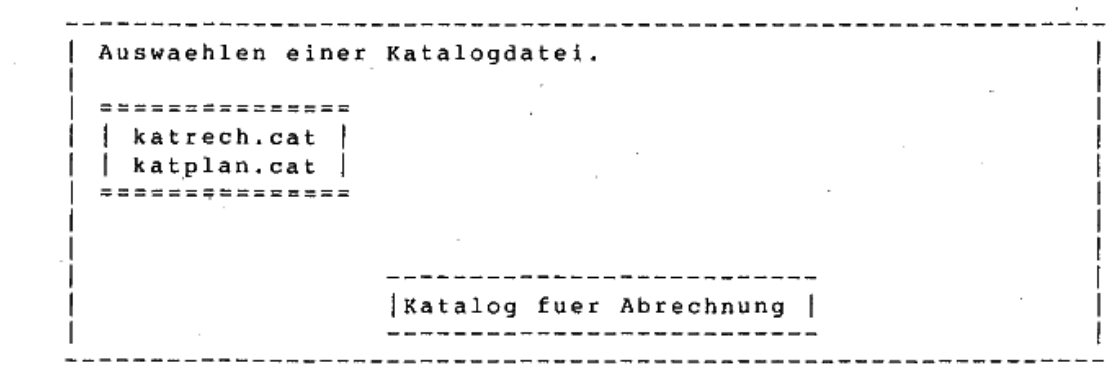



Bild 95 Eroeffnung einer Katalogdatei mit SET KATALOG TO ?

Mit den Abwaerts- oder Aufwaerts-Pfeiltasten stellen Sie Ihre gewuenschte Katalogdatei ein und loesen die Eroeffnung mit <ET> aus. Die Katalogdatei muss sich allerdings im gleichen Verzeichnis wie die Hauptkatalogdatei befinden, naemlich im aktuellen Verzeichnis des aktuellen Laufwerks (siehe auch SET DEFAULT).

Andernfalls wird nach bejahter Frage zur Erzeugung einer neuen Katalogdatei im aktuellen Verzeichnis eine Katalogdatei mit vorgegebenen Namen angelegt und nur das Pfadfeld in der Hauptkatalogdatei aktualisiert (s. Abschnitt 6.3.).

Sind in der Hauptkatalogdatei CATALOG.CAT Katalogdateien eingetragen, die nicht auf den im Zugriff stehenden Datentraegern existieren, werden diese Eintraege in der Hauptkatalogdatei automatisch geloescht.

Die zweite Art der Eroeffnung bietet sich dann an, wenn Ihnen die Namen Ihrer Katalogdateien entfallen sind. Wenn Sie die Auswahl einer Katalogdatei unterbinden wollen, geben Sie <Esc>.

Beachten Sie bitte, dass bei der Arbeit mit einer Katalogdatei der Arbeitsbereich 10 nicht fuer andere Dateien belegt werden kann und 2 von den maximal 15 gleichzeitig eroeffneten Dateien fuer die Katalogdatei und CATALOG.CAT abgehen!
Die Eroeffnung einer Katalogdatei zieht den Befehl SET CATALOG ON nach sich und setzt damit bei den im Abschnitt 6.2. aufgefuehrten Befehlen eine automatische Aktualisierung der Katalogdatei in Kraft.
Moechten Sie die Katalogdatei eroeffnet lassen, aber die automatische Aktualisierung unterdruecken, geben Sie nur SET CATALOG OFF ein.

Eine eroeffnete Katalogdatei koennen Sie mit SET CATALOG TO schliessen. Dabei wird SET CATALOG durch das System auf OFF gesetzt.

## 6.2. Automatische Aktualisierung von Eintragungen in einer Katalogdatei

Eine geoeffnete Katalogdatei wird bei einer Reihe von Befehlen automatisch aktualisiert, wenn nicht vorher SET CATALOG OFF gesetzt wurde. So wird ein Hinzufuegen von Eintragungen bei folgenden Befehlen gewaehrleistet:

```
COPY STRUCTURE TO
COPY TO ... STRUCTURE EXTENDED
COPY TO
CREATE
CREATE ... FROM
CREATE/MODIFY LABEL
CREATE/MODIFY QUERY
CREATE/MODIFY REPORT
CREATE/MODIFY SCREEN
CREATE/MODIFY VIEW
IMPORT FROM
INDEX
JOIN
SET FORMAT
SET INDEX
SET VIEW
SORT
TOTAL
USE
```

Mit Ausnahme der Befehle INDEX und SET INDEX, sowie der MODIFY-Befehle (wenn die zu modifizierende Datei bereits Bestandteil der Katalogdatei ist) erkennen Sie die Aufnahme einer Eintragung in die Katalogdatei an der Aufforderung, einen Titel einzugeben. Natuerlich darf die Voreinstellung SET TITLE ON nicht veraendert sein.
Das folgende Beispiel demonstriert die Eintragung der Datenbankdatei "auftrag" und der Indexdateien "kunrauf" und "aufnrau" in die Katalogdatei "katrech".

```
SET CATALOG TO katrech
USE auftrag INDEX kunrauf,aufnrau
Eingeben eines Titels fuer Datei auftrag.dbd:
Auftragsdatei vom 4.9.89
```

Bringen Sie auf die gleiche Weise die Datenbankdatei "kundenl" mit dem Titel "Kundendatei mit Merkfeldern" in die Katalogdatei!

Bei Indexdateien wird automatisch der Indexausdruck im Titelfeld der Katalogdatei eingetragen.

CREATE/MODIFY - Befehle, die eine geoeffnete Datenbankdatei oder andere geoeffnete Dateien voraussetzen, erfordern fuer eine Katalogfuehrung, dass die vorausgesetzten Dateien ebenfalls in der Katalogdatei eingetragen sind. Sonst erscheint die Nachricht

```
Datei nicht im Katalog.
SET CATALOG war OFF bei Benutzung der Datenbankdatei.
```

Der Befehl ERASE unterstuetzt automatisch das Loeschen von Eintragungen in einer geoeffneten Katalogdatei. Bitte beachten Sie dabei, dass die im Befehl angegebene Datei nicht nur als Eintrag in der Katalogdatei, sondern selbst auf der Diskette oder der Festplatte geloescht wird!

Das automatische Aendern eines Dateinamens wird beim RENAME-Befehl sowohl im entsprechenden Verzeichnis als auch in einer geoeffneten Katalogdatei vollzogen.

## 6.3. Die Struktur von Katalogdateien

Sowohl die Hauptkatalogdateien CATALOG.CAT als auch die vom Nutzer angelegten Katalogdateien sind Datenbankdateien mit der im Bild 96 dargestellten Struktur.

```
Datenbankdateistruktur : a:catalog.cat
Anzahl der Datensaetze :        2
Letztes Aenderungsdatum: 04.09.89
Feld     Feldname     Typ          Laenge    Dez
   1     PATH         Zeichen          70
   2     FILE_NAME    Zeichen          12
   3     ALIAS        Zeichen           8
   4     TYPE         Zeichen           3
   5     TITLE        Zeichen          80
   6     CODE         Numerisch         3
   7     TAG          Zeichen           4
* Gesamt *                           181
```

Bild 96 Struktur einer Katalogdatei

Bei der automatischen Aktualisierung (s. Abschnitt 6.2.) sorgt REDABAS-4 fuer die ordnungsgemaesse Eintragung in den Feldern. Zur Eingabe des Feldes 5, Titel (TITLE), werden Sie jeweils bei einem Zugang einer Datei aufgefordert (bei Voreinstellung SET

TITLE ON). Das Feld 7, Kennzeichen (TAG), wird von REDABAS-4 nicht belegt und kann vom Nutzer verwendet werden.

Die Felder haben folgende Bedeutung:

- Feld 1, Pfad (PATH)
  Es wird der zum Zeitpunkt der Aktualisierung angegebene Pfad einschliesslich Dateiname und Dateityp abgespeichert. Die in der Katalogdatei eingetragenen Dateien werden ueber diesen Pfad aufgerufen, wenn sie aufgrund eines Befehls mit einer Katalog-Frage-Klausel (s. Abschnitt 6.5.) auszuwaehlen sind. Katalogdateien, die nicht im aktuellen Verzeichnis angelegt sind, lassen sich nicht nach einem SET CATALOG TO ? - Befehl auswaehlen.

- Feld 2, Dateiname (FILE_NAME)
  Dieses Feld enthaelt Dateiname und Dateityp (entspricht Feld 1 ohne Pfadangaben).

- Feld 3, Alias (ALIAS)
  Dieses Feld nimmt den Dateinamen ohne Dateityp auf.

- Feld 4, Typ (TYPE)
  Dieses Feld enthaelt den fuer den Dateityp festgelegten Voreinstellwert der Erweiterung, unabhaengig davon, ob Sie eine andere Erweiterung verwenden, z.B. wird nach USE kunden.adr fuer die Datenbankdatei "kunden.adr" der Typ "dbd" eingetragen.

- Feld 5, Titel (TITLE)
  Die Eingabe dieses Feldes wird bei einem Neuzugang mit Ausnahme einer Indexdatei angefordert. Bei Indexdateien wird der fuer die Indizierung festgelegte Ausdruck uebernommen.

- Feld 6, Code (CODE)
  Dieses Feld ist fuer einen Code zur Kennzeichnung der Zusammengehoerigkeit von Dateien bestimmt. Katalog- und Viewdatei werden durch 0 gekennzeichnet. Der Code wird fortlaufend ab 1 vergeben. Dateien, die aus derselben Datenbankdatei erstellt werden, erhalten denselben Code.

- Feld 7, Kennzeichen (TAG)
  Dieses Feld wird fuer die Katalogfuehrung nicht benutzt und kann von Ihnen fuer Kommentare verwendet werden.

## 6.4. Manipulation in Katalogdateien

Zusaetzlich zur automatischen Aktualisierung einer Katalogdatei (s. Abschn. 6.2.) haben Sie die Moeglichkeit, direkte Manipulationen an der Katalogdatei vorzunehmen.
Da Katalogdateien normale Datenbankdateien sind, koennen die Funktionen der Datenbankverwaltung unter Beachtung der im Abschnitt 6.3. erklaerten Struktur ausgefuehrt werden. Insbesondere koennen Sie Eintraege in Katalogdateien hinzufuegen, loeschen oder aendern (z.B. mit BROWSE) ohne die eigentlichen Dateien zu beruehren. Beispielsweise bleibt im Gegensatz zu ERASE (vgl. Abschnitt 6.2.) eine Datei auf dem Datentraeger bestehen, wenn ihr Eintrag in einer Katalogdatei geloescht

wird.

Wir moechten Sie jedoch beim direkten Aendern in einer Katalogdatei warnen, da leicht der innere Zusammenhang der Katalogeintragung und die Widerspiegelung der Dateienbelegung auf dem Datentraeger verloren gehen kann. Voellig ohne Komplikationen koennen Sie jedoch die Feldinhalte fuer Titel (TITLE) und Kennzeichen (TAG) aendern.

Voraussetzung fuer die Manipulation in einer Katalogdatei bleibt, dass die Katalogdatei eroeffnet ist. Eine Hauptkatalogdatei wird mit dem Befehl

```
USE catalog.cat
```

eroeffnet. Katalogdateien, die noch nicht im Arbeitsbereich 10 eroeffnet sind, lassen sich ebenfalls mit

```
USE <katalogdatei>.CAT
```

eroeffnen.

Wir weisen Sie darauf hin, bei der Arbeit mit einer Katalogdatei waehrend der Manipulation einer anderen Katalogdatei in einem Arbeitsbereich ungleich 10 SET CATALOG OFF zu geben, da andernfalls Ihre manipulierte Katalogdatei als Datenbankdatei katalogisiert wird.

Fuer eine durch SET CATALOG TO <katalogdatei> im Arbeitsbereich 10 eroeffnete Katalogdatei muss erst SELECT 10 gegeben werden, bevor Manipulationsbefehle folgen duerfen.

## 6.5. Die Katalog-Frage-Klausel

Bei einer Reihe von Befehlen gestattet die Syntax anstelle einer Datei ein Fragezeichen, die sogenannte Katalog-Frage-Klausel, zu spezifizieren. Laufwerk- und Pfadangaben sind dabei nicht moeglich. Anstelle der Ausfuehrung der fuer eine Datei vorgesehenen Funktion wird eine Liste aller der in einer geoeffneten Katalogdatei registrierten Dateien angezeigt, die fuer die Funktion relevant sind. Bei Befehlen, die sich auf eine geoeffnete Datenbankdatei beziehen, werden in der Liste nur die Dateien aus der Katalogdatei angezeigt, die in Verbindung mit dieser Datenbankdatei definiert wurden.
Die Liste der Dateien wird in einem Doppelrahmen in Form eines Menues angeboten. In einem Einfachrahmen erscheint der Titel der voreingestellten Datei als Kommentar. Mittels der Unten- und Oben-Pfeiltasten und nachfolgendem <ET> kann die Datei ausgewaehlt werden, fuer die die vorgesehene Funktion nun realisiert wird.

Mit <Esc> koennen Sie eine irrtuemlich ausgewaehlte Funktion verwerfen und in Ihren urspruenglichen Modus, also in den interaktiven Modus oder Programmodus, zurueckkehren.

Katalog-Frage-Klauseln sind auf folgende Befehle anwendbar:

```
CREATE/MODIFY LABEL  ?
CREATE/MODIFY QUERY  ?
CREATE/MODIFY REPORT ?
CREATE/MODIFY SCREEN ?
CREATE/MODIFY VIEW   ?
ERASE ?
LABEL FORM ?
REPORT FORM ?
SET CATALOG TO ?
SET FILTER TO FILE ?
SET FORMAT TO ?
SET INDEX TO ?
SET VIEW TO ?
USE ?
```

Wir moechten die Benutzung der Katalog-Frage-Klausel am USE ?-Befehl erklaeren. Unter der Voraussetzung, dass Sie die Befehle

```
SET CATALOG TO katrech
USE ?
```

eingegeben haben, erscheint am Bildschirm im Doppelrahmen die Liste der in der Katalogdatei "katrech" eingetragenen Datenbankdateien (vgl. Bild 97).

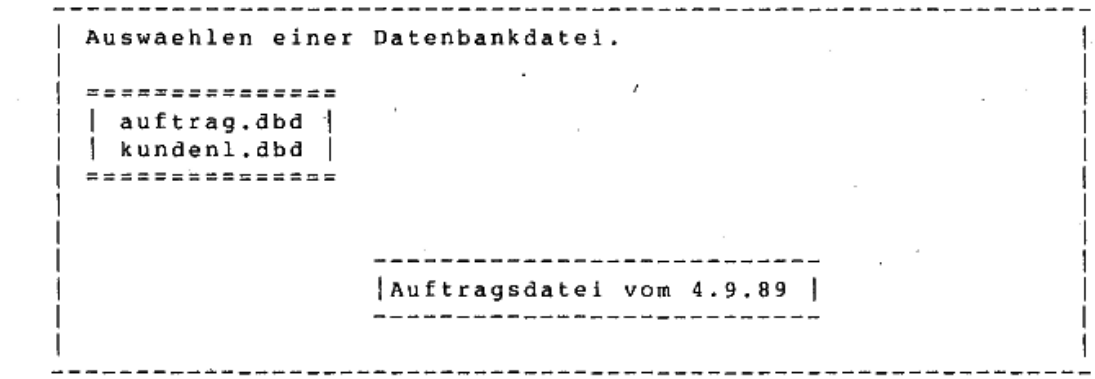



Bild 97 Eroeffnung einer Datenbankdatei mit USE ?

Im Einfachrahmen ist der Titel der voreingestellten Datenbankdatei sichtbar. Betaetigen Sie <ET>, um "auftrag" zu oeffnen! Es wird im Doppelrahmen eine Liste der zu "auftrag" zugehoerigen Indexdateien angezeigt, die in der Katalogdatei eingetragen sind (vgl. Bild 98).

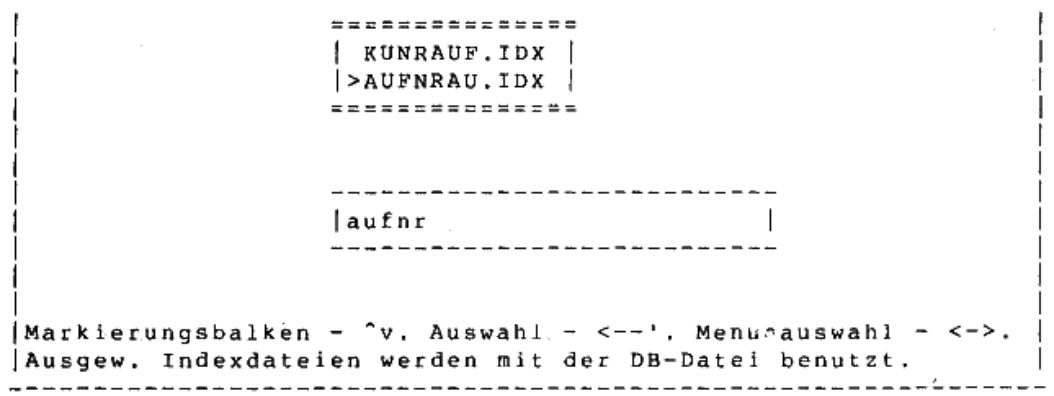



Bild 98 Auswahl zur Datenbankdatei gehoeriger Indexdateien

Im Einfachrahmen wird der zur voreingestellten Indexdatei zugehoerige Schluesselausdruck ausgewiesen. Stellen Sie den Markierungsbalken mit der Unten-Pfeiltaste auf

AUFNRAU.IDX

ein und waehlen Sie diese Indexdatei mit <ET> aus!

Gehen Sie mit der Links-Pfeiltaste zur naechsten Befehlseingabe ueber.
Mit der <F6>-Taste koennen Sie das Auswahlergebnis ueberpruefen. Es wird u.a. angezeigt:

```
 : display status

Selektierter Arbeitsbereich
Arbeitsbereich: 1, Datenbankdatei: A:auftrag.dbd Alias:AUFTRAG
       Haupt-Indexdatei: A:aufnrau.idx Schluessel: aufnr

Arbeitsbereich:10, Datenbankdatei: A:katrech.cat Alias:CATALOG.
```

## 7. Umsteigen von REDABAS auf REDABAS-4

REDABAS-3 und REDABAS-4 sind aufwaertskompatibel, d. h. saemtliche unter REDABAS-3 erstellte Anwendungsprogramme sind auch unter REDABAS-4 lauffaehig. Es gibt keine Veraenderungen in der Struktur der REDABAS-3 Dateien.

Obwohl ein grosser Teil der REDABAS-Nutzer bereits den Umstieg von REDABAS auf DEDABAS-3 vollzogen hat, gibt es aufgrund der Vielzahl von existierenden REDABAS-Anwendungen fuer 8- und 16-Bit-Mikrorechner unter SCP bzw. SCP1700 eine Zahl von Anwendern, die direkt von REDABAS auf REDABAS-4 umsteigen wollen. Deswegen soll auf die Fragen der Kompatibilitaet beider Systeme hier naeher eingegangen werden.

Beim Umstieg von REDABAS auf REDABAS-4 sind zunaechst die unterschiedlichen Diskettenformate zu beachten, die von den Betriebssystemen SCP bzw. SCP1700 einerseits und DCP andererseits benutzt werden. Durch die neueren Betriebssystemversionen des DCP erfolgt hier eine recht gute Unterstuetzung. Bereits beim Systemstart kann ein SCP-Diskettentreiber eingebunden werden, der die direkte Verarbeitung von SCP-Diskettenformaten unter DCP ermoeglicht.

Gewoehnlich wird einem physischen Laufwerk (z.B. A:) ein logischer Laufwerksname (z.B. F:) zugeordnet, unter dem eine SCP-Diskette angesprochen werden kann. Die von einem Standardformat abweichenden, allgemein gebraeuchlichen SCP-Diskettenformate lassen sich durch ein Menueprogramm (SCPSET) voreinstellen.

Da REDABAS-4 nicht schlechthin eine Weiterfuehrung von REDABAS, sondern eine Neuentwicklung darstellt, in der neben einer Reihe neuer Befehle und Funktionen auch erhebliche Unterschiede in der Syntax und im Aufbau der REDABAS-Dateien zu verzeichnen sind, ergibt sich, dass eine bereits existierende REDABAS-Anwendung (bestehend aus Daten und Anwendungsprogrammen) nicht direkt unter REDABAS-4 lauffaehig ist.
Jedoch wird dem Nutzer unter dem Namen RETRANS ein Programm angeboten, das die Transformation von REDABAS-Dateien weitgehend unterstuetzt und folgende Aspekte beruecksichtigt:

- Aenderung von Namen und der Syntax von Befehlen und Funktionen
- Aenderungen in der Wirkungsweise einiger Befehle und Funktionen
- Aenderungen (Erweiterungen) in der Datenbankstruktur.

Dem Benutzer von RETRANS steht eine "Programmtechnische Beschreibung" zur Verfuegung, in der die Anwendung dieses Transformationsprogramms detailliert beschrieben ist.

## 7.1. Transformation von Datenbankdateien (.DBD)

Die Erweiterung der Leistungsparameter von REDABAS-4 betreffs der Datenbankdateien (Feldanzahl je Satz, Satzanzahl je Datei, Anzahl der Datentypen) fuehrte zu einem geaenderten Aufbau der Strukturbeschreibung, die sich am Anfang jeder Datenbankdatei befindet. Diese Transformation nimmt RETRANS automatisch vor.

Zu beachten ist, dass REDABAS-4 im Feldnamen keinen Doppelpunkt ":" erlaubt. Hiefuer setzt RETRANS einen Unterstrich "_".

Bei Datenbankdateien (und nur bei diesen) wird eine Ruecktransformation vom REDABAS-4 in das REDABAS-Format unterstuetzt, wenn die REDABAS-Bedingungen eingehalten werden (Feldanzahl je Datensatz max. 32, Satzlaenge max. 1000 Bytes, Satzanzahl je Datei max. 65535). Felder vom Datentyp MERK sind nicht zulaessig, Felder vom Datentyp (D) werden jedoch verarbeitet und in den Datentyp "C" (Zeichenreihe) umgewandelt.

## 7.2. Transformation von Indexdateien (.IDX)

RETRANS transformiert Indexdateien nicht direkt, sie koennen gewoehnlich mit geringem manuellen Aufwand neu erstellt werden. Wird trotzdem RETRANS dazu benutzt, ermittelt das Transformationsprogramm den Indexausdruck aus der alten Indexdatei und erstellt eine Programmdatei, die nur den notwendigen INDEX-Befehl zum Erzeugen der neuen Indexdatei enthaelt. Unter REDABAS-4 ist dann die Datenbankdatei mit USE zu eroeffnen und die durch RETRANS erzeugte Programmdatei abzuarbeiten.

## 7.3. Transformation von Programm- und Maskendateien (.PRG,.MSK)

In Befehlsfolgen kann RETRANS nur dort eine automatische Transformation vornehmen, wo eine formelle Aenderung in der Syntax zwischen REDABAS und REDABAS-4 erfolgte. Aufgrund des erweiterten Befehls- und Funktionsumfanges und der hoeheren Leistung von REDABAS-4 (z.B. paralleler Zugriff auf 10 Datenbankdateien) koennte bei vielen Anwendungsprogrammen der Algorithmus effektiver gestaltet werden. Diese manuelle Arbeit bleibt dem Anwendungsprogrammierer vorbehalten.

Bei Befehlen, die sich in ihrer Wirkung zwischen REDABAS und REDABAS-4 unterscheiden, setzt RETRANS in die transformierte Datei vor die betreffende Befehlszeile eine Kommentarzeile mit Hinweisen fuer den Programmierer ein.

Am Anfang jeder Befehlsdatei fuegt RETRANS die Befehle SET HEADING OFF (Unterdrueckung der Feldnamenanzeige bei LIST und DISPLAY) und SET SAFETY OFF (automatisches Ueberschreiben von Dateien) ein, um eine zu REDABAS analoge Wirkung zu erzielen. Bei Bedarf koennen diese Zeilen natuerlich wieder geloescht werden, und es gilt die Standardzuweisung (ON).

Im folgenden werden einige formelle Aenderungen angefuehrt, die RETRANS selbstaendig ausfuehrt:

- Ersetzen von Funktionen und Befehlen mit neuem Namen

REDABAS

```
ERASE
USE bestand
LOCATE FOR !(bezei)="BOLZEN"
IF *
? "Satznr.",#,"geloescht!"
ENDIF
```

REDABAS-4

```
CLEAR
USE bestand
LOCATE FOR UPPER(bezei)="BOLZEN"
IF DELETED()
? "Satznr.",RECNO(),"geloescht!"
ENDIF
```

- Begrenzen der Wahrheitswerte (T,J,N,F) durch Punkte (Vorsicht! Wenn diese Buchstaben als Variablennamen benutzt werden, erfolgt u.U. auch die Punkteingrenzung und macht eine manuelle Rueckaenderung notwendig!)

- Ersetzen des Doppelpunktes ":' im Namen von Speichervariablen und Feldern durch den Unterstrich "_".

REDABAS

```
STORE T    TO v1
STORE n    TO var:1
STORE 123 TO n
? n
STORE 123 TO n1
? n1
```

REDABAS-4

```
STORE .T. TO v1
STORE .n. TO var_1
STORE 123 TO n
? .n.
STORE 123 TO n1
? n1
```

- Ersetzen der Arbeitsbereiche PRIMARY bzw. SECONDARY und der Vorsatzzeichen P. bzw. S. bei Feldnamen durch die Standard-Aliasnamen A bzw. B und Einfuegen des Zeigersymbols "->" (zu beachten ist hier, dass REDABAS die Vorsatzzeichen nur verlangt, wenn gleiche Feldnamen in beiden Arbeitsbereichen vorkommen, dass RETRANS die Vorsatzzeichen nur bei Vorhandensein transformieren kann und dass REDABAS-4 unbedingt den Aliasverweis vor dem Feldnamen erwartet, wenn es sich um Felder eines nichtaktuellen Arbeitsbereiches handelt.)

REDABAS

```
SELECT PRIMARY
USE kunden
SELECT SECONDARY
USE auftrag
LOCATE FOR S.kunr=P.kunr
?name,p.kunr,s.betrag,beza
```

REDABAS-4

```
SELECT A
USE kunden
SELECT B
USE auftrag
LOCATE FOR B->kunr=A->kunr
?name,A->kunr,B->betrag,beza
```

Bei Abarbeitung der transformierten Befehlsfolge unter REDABAS-4 wuerde im letzten Befehl die Variable "name" nicht gefunden werden, da es sich um einen Feldnamen der Datei "kunden" handelt und der Aliasverweis fehlt.

Neben den formellen Aenderungen, die RETRANS automatisch durchfuehrt (teilweise mit zusaetzlichen Kommentaren), gibt es eine

Reihe von Funktionen und Befehlen in REDABAS-4, die sich in der Wirkungsweise von REDABAS unterscheiden und auf die RETRANS nur teilweise durch Kommentare hinweisen kann. Einige dieser Unterschiede werden im folgenden genannt. Beachten Sie bitte bei der Abarbeitung der nachfolgenden Beispiele, dass die mit ":" gekennzeichneten Befehlszeilen im interaktiven Modus, nichtgekennzeichnete Befehlszeilen im Programmodus einzugeben sind.

- Einstellen des Satzzeigers
  REDABAS setzt die Satznummer auf 0, wenn die Suche mit FIND erfolglos ist, REDABAS-4 dagegen stellt die Satznummer auf die maximale Satzanzahl +1 und setzt die EOF()-Funktion auf "wahr" . Die Satznummer 0 wird von REDABAS-4 nicht benutzt, dafuer gibt es die neue Dateianfang-Funktion BOF().

| REDABAS | REDABAS-4 |
|---|---|
| : USE kunden | : USE kunden |
| : INDEX ON kunr TO ikunr | : INDEX ON kunr TO ikunr |
| 00013 SAETZE INDIZIERT | 100% indiziert 13 Saetze indiziert |
| : FIND 777 | : FIND 777 |
| SCHLUESSEL ODER ZEICHENKETTE NICHT VORHANDEN! | Nicht gefunden. |
| : ? # | : ? RECNO() |
| 0 | 14 |
| | : ? EOF() |
| | .T. |
| | : GO TOP |
| | : ? BOF() |
| | .F. |
| | : SKIP-1 |
| | Satz-Nr. 1 |
| | : ? BOF() |
| | .T. |

- Zugriff auf einen Datensatz in einer zweiten Datei.
  Bei REDABAS kann zwar das automatische Wiederauffinden eines Datensatzes aus einer zweiten Datei in Abhaengigkeit eines aus einer ersten Datei gelesenen Satzes ueber SET LINKAGE ON erfolgen, aber die Verbindung ist nur ueber die Satznummer moeglich, deshalb wird dafuer meist der Befehl FIND genutzt und der Suchbegriff ueber eine Speichervariable vermittelt.

  In REDABAS-4 erlaubt der SEEK-Befehl direkt die Angabe eines Ausdruckes, der den Suchbegriff enthaelt. Noch einfacher ist es jedoch mit REDABAS-4 ueber SET RELATION TO <ausdr> INTO <alias> das automatische Wiederauffinden des Datensatzes aus der zweiten Datei anhand des gewuenschten Ordnungsbegriffes zu veranlassen.

| REDABAS | REDABAS-4 |
|---|---|
| SET TALK OFF | SET TALK OFF |
| SELECT SECONDARY | SELECT 2 |
| USE kunden INDEX ikunr | USE kunden ALIAS ku INDEX ikunr |
| SELECT PRIMARY | SELECT 1 |
| USE auftrag | USE auftrag |
| DO WHILE .NOT. EOF | DO WHILE .NOT. EOF() |

```
STORE kunr TO vkunr
SELECT SECONDARY
FIND &vkunr
? kunr,name,betrag
SELECT PRIMARY
SKIP
ENDDO
SET TALK ON
```

```
SELECT 2
SEEK A->kunr
? kunr,name,A->betrag
SELECT 1
SKIP
ENDDO
SET TALK ON
```

oder

```
SET TALK OFF
SELECT 2
USE kunden ALIAS ku INDEX ikunr
SELECT 1
USE auftrag
SET RELATION TO kunr INTO ku
DO WHILE .NOT. EOF()
? kunr, ku->name, betrag
SKIP
ENDDO
SET TALK ON
```

- Zugriff auf Speichervariablen

In REDABAS sind alle Speichervariablen global gueltig, d. h. unabhaengig davon, in welcher Programmebene sie erklaert wurden, sind sie immer ansprechbar, solange sie nicht durch einen der Befehle CLEAR, RELEASE oder RESTORE geloescht werden.
Bei REDABAS-4 sind standardmaessig nur die im interaktiven Modus definierten Variablen global gueltig, die in einem Programm oder Unterprogramm definierten Variablen haben nur lokale Gueltigkeit und werden bei Beendigung des Programmes geloescht. Damit sind die in einem Programm definierten Variablen fuer diesem Programm untergeordnete Programme verfuegbar, fuer uebergeordnete Programme jedoch nicht.

Soll eine Variable ueber das Programmende hinaus erhalten bleiben, wird sie mit dem PUBLIC-Befehl als global gueltig erklaert. Wenn der Programmierer eines Unterprogrammes fuer seine lokal gueltigen Variablen gleiche Namen benutzen will wie sie bereits im Rahmenprogramm fuer andere Variablen vergeben wurden oder die im uebergeordneten Programmen verwendeten Namen von Speichervariablen ihm nicht bekannt sind, so kann er diese durch den PRIVATE-Befehl schuetzen. Solche -als privat erklaerte- Speichervariablen werden bei einer Wertezuweisung im Unterprogramm ein zweites Mal angelegt, wenn der Name bereits im uebergeordneten Programm existiert. REDABAS-4 loescht diese Variable wieder bei Beendigung des Unterprogramms.

```
* p1.prg
SET TALK OFF
v0='zzz'
v1='abc'
DO p2
? v1
? v2
? v3
SET TALK ON
RETURN
```

```
* p2.prg
PRIVATE v1
PUBLIC v2
v1=123
v2='def'
v3='ghi'
DO p3
? v4
RETURN
```

```
* p3.prg
v4=44
? v0
? v1
? v2
? v3
? v4
RETURN
```

```
: DO p1
zzz
     123
def
ghi
      44
Variable nicht gefunden.
     ?
? v4
Aufgerufen von - A:p2.prg
Aufgerufen von - A:p1.prg
Abbruch, Ignorieren, Unterbrechen ? (A,I,U) I
abc
def
Variable nicht gefunden.
     ?
? v3
Aufgerufen von - A:p1.prg
Abbruch, Ignorieren, Unterbrechen ? (A,I,U) I
: LIST MEMORY
v2    pub    C    "def"
  1 Speichervariable def., 5 Bytes benutzt
  .
  .
  .
```

- Nullwert in einer Speichervariablen

  Wird bei einer Eingabeaufforderung fuer eine Speichervariable nur die <ET>-Taste gedrueckt, belegt REDABAS diese Variable mit einem Leerzeichen. REDABAS-4 tut das nicht, setzt aber die Laenge der Variablen auf Null. Beim Test einer Konsoleingabe ist dies zu beachten. In diesem Zusammenhang ist auch eine Aenderung in der TRIM()-Funktion zu sehen. Waehrend bei REDABAS die TRIM()-Funktion -auf eine Folge von Leerzeichen angewendet- ein einzelnes Leerzeichen ergibt, erzeugt REDABAS-4 in diesem Fall eine Zeichenreihe der Laenge 0. (Uebrigens fuegt REDABAS-4 bei ACCEPT und INPUT nicht automatisch den Doppelpunkt an !)

REDABAS

```
* null.prg
ACCEPT "Eingabe" TO sein
IF sein=' '
? 'Nur <ET> gedrueckt!'
? LEN(sein)
ELSE
? sein
ENDIF
```

```
: DO null
Eingabe: <ET>
Nur <ET> gedrueckt
1
: USE kunden
: APPEND BLANK
: ? LEN(vorname)
8
: ? LEN(TRIM(vorname))
1
```

REDABAS-4

```
* null.prg
ACCEPT "Eingabe?" TO sein
IF LEN(sein)=0
? 'Nur <ET> gedrueckt!'
ELSE
? sein
ENDIF
```

```
: DO null
Eingabe? <ET>
Nur <ET> gedrueckt!

: USE kunden
: APPEND BLANK
: ? LEN(vorname)
8
: ? LEN(TRIM(vorname))
0
```

- Rechengenauigkeit und Runden

  Bei REDABAS richtet sich die angezeigte Genauigkeit eines Rechenergebnisses nach der Anzahl der Dezimalstellen der Ausgangsgroessen. Weitere Dezimalstellen werden abgeschnitten, es erfolgt keine Rundung. Bei REDABAS-4 kann die standardmaessige Genauigkeit von zwei Dezimalstellen durch SET DECIMALS TO <ausdrN> veraendert werden, ausserdem wird das Ergebnis gerundet bzw. es steht die neue ROUND()-Funktion zur Verfuegung.
  Zu beachten ist, dass in der ROUND()-Funktion die gewuenschte Stellenzahl anzugeben ist, die Stellenzahl der Anzeige des Ergebnisses aber durch SET DECIMALS ... bestimmt wird.

```
        REDABAS                              REDABAS-4
        -------                              ---------
: ? 6/7                              : ? 6/7
0                                    0.86
: ? 6.00/7                           : SET DECIMALS TO 4
0.85                                 : ? 6/7
: ? 6.000/7                          0.8571
0.857                                : SET DECIMALS TO 8
:?INT((6/7+0.005)*100)/100.00        : ? ROUND (6/7,6)
0.86                                 0.85714300
```

- Test eines Eingabewertes auf Gueltigkeit

  Bei der Eingabe von Werten ueber den @-Befehl ist oft ein Test zur Einhaltung eines Wertebereiches wuenschenswert. Mit REDABAS ist dies nur ueber eine Programmschleife moeglich. Wenn mehrere solcher Eingabefelder in einer Bildschirmmaske enthalten sind, kann die notwendige Befehlsfolge aufwendig sein. REDABAS-4 erlaubt mit der RANGE-Klausel im @-Befehl die Vorgabe eines Wertebereiches und erzeugt bei Nichteinhaltung desselben ein akustisches Signal und eine Fehlermeldung.

```
        REDABAS
        -------
      .
      .
      .
STORE 0 TO v1
DO WHILE v1<1.0 .OR. v1>3.9
@ 3,5 SAY 'Faktor' GET v1 PICTURE '9.9'
READ
ENDDO
      .
      .
      .
```

```
        REDABAS-4
        ---------
             .
             .
             .
v1=0
@ 3,5 SAY 'Faktor' GET v1 PICTURE '9.9' RANGE 1.0, 3.9
READ
             .
             .
             .
```

## 7.4. Transformation von Speichervariablen- und Reportdateien (.VAR, .DEF)

Speichervariablen- und Reportdateien lassen sich problemlos transformieren. Wenn im Namen von Variablen Doppelpunkte auftreten, ersetzt RETRANS diese durch den Unterstrich. RETRANS kann natuerlich keine Unterscheidung zwischen den Attributen PUBLIC und PRIVATE treffen. Wenn solch eine Differenzierung unter REDABAS-4 nicht erforderlich ist, werden am besten alle Variablen im Hauptprogramm oder in der Befehlsebene erstellt (RESTORE-Befehl). Bei der Transformation von Reportdateien ist zu beachten, dass unter REDABAS-4 diese Dateien kein Textformat besitzen.

Ausserdem sei daran erinnert, dass die unter REDABAS moeglichen Voreinstellungen fuer eine zusaetzliche Ueberschrift (SET HEADING TO...) und den Seitenvorschub vor dem Druck (SET EJECT ON/OFF) nicht mehr in dieser Form verfuegbar sind, sondern unter REDABAS-4 direkt als Klausel (HEADING bzw. NOEJECT) im REPORT-Befehl enthalten sind. Funktionsnamen, die im Spalteninhalt auftreten koennen, transformiert RETRANS automatisch.

# 8. Allgemeine Informationen

## 8.1. Datenorganisation

### 8.1.1. Aufbau der Datenbank

Unter dem Begriff "Datenbank" ist die vom Computer gespeicherte Datenmenge zu einem betrachteten Gegenstandsbereich zu verstehen. Dazu gehoeren nicht nur die den Anwender interessierenden Datenelemente selbst, sondern auch die Beziehungen zwischen ihnen sowie alle gespeicherten Verwaltungs- und Zugriffshilfen. Das "Datenbankbetriebssystem" (Datenbankverwaltungssystem) hat die Aufgabe, das Aufbauen, Verwalten und Nutzen der Datenbank zu unterstuetzen, vor allem durch Bereitstellen einer geeigneten "Datenbanksprache". Es kann mehrere nach gleichem Prinzip aufgebaute Datenbanken bedienen.
REDABAS-4 ist ein "relationales" Datenbankbetriebssystem, denn es strukturiert die Daten der Datenbank entsprechend dem relationalen Datenmodell.

REDABAS-4 organisiert die Datenmenge in Form von Tabellen und kommt damit den Vorstellungen des Anwenders entgegen. Eine zweidimensionale Tabelle wird durch eine Relation realisiert, die im traditionellen Sinn einer Datei entspricht. Die Spalten der Tabelle werden durch die Datenfelder verkoerpert, die Zeilen der Tabelle entsprechen den Datensaetzen.

| Teile-Nr. | Bezeichnung | Anzahl | Preis | Lagerort |
|---|---|---|---|---|
| 00819 | Bremsleitung | 31 | 5.80 | 140 |
| 00852 | Bolzen | 471 | 0.10 | 110 |
| 14043 | Ansaugkruemmer | 194 | 2.45 | 140 |
| 17000 | Blinkleuchte, vorn | 114 | 10.00 | 130 |

Bild 99 Tabelle eines Lagerbestandes

Die Informationen zum Tabellenaufbau, d.h. Spaltenbezeichnung (Feld-Name), Datentyp (Zeichen, numerisch, logisch, Datum, MERK) und Spaltenbreite (Feldlaenge) hinterlegt REDABAS-4 im Kopf der Datenbankdatei, bevor die Speicherung der Tabellenwerte erfolgt. Bei der Erfassung einer Tabelle speichert REDABAS-4 die Zeilen als Datensaetze sequentiell in der Eingabereihenfolge ab.

REDABAS-4 kann beliebig viele Tabellen als Datenbankdateien speichern. Die interne Darstellung der Feldwerte entspricht weitgehend der externen; die ASCII-Zeichen werden nicht konvertiert.

Der Platzbedarf einer Datenbankdatei auf Diskette (in Bytes) errechnet sich aus:

Satzanzahl * Satzlaenge + 32 * Feldanzahl + 34

Es ist zu beachten, dass der tatsaechliche Platzbedarf etwas groesser als die errechnete Laenge ist, da das Betriebssystem DCP den Speicherplatz auf der Diskette oder Festplatte in

Portionen fester Laenge vergibt, die einem Vielfachen der Sektorlaenge 512 entsprechen.

REDABAS-4 verlangt fuer die Datensaetze kein Sortierkriterium. Wuenscht der Nutzer jedoch eine bestimmte Reihenfolge der Datensaetze, so kann er eine Sortierung oder Indizierung der Datenbankdatei herbeifuehren (vgl. Abschnitt 3.4. und 9.3.3.). Bei einer Sortierung (mit dem Befehl SORT) erfolgt eine physische Umordnung der Datensaetze. Bei Ergaenzung der Datenbankdatei muss der Nutzer fuer die Einhaltung der Sortierordnung selbst Sorge tragen, indem er die Position fuer die neuen Saetze bestimmt, oder es muss ein neuer Sortiervorgang eingeleitet werden.

Eine zweite Methode zum Erreichen einer sortierten Bereitstellung bzw. Verarbeitung der Datensaetze ist die Indizierung (mit dem Befehl INDEX). Bei der Indizierung wird eine zusaetzliche Datei erstellt, die zu jedem Datensatz den Schluesselbegriff und einen Zeiger zum Datensatz enthaelt. Solche Indexdateien erlauben neben einer Satzverarbeitung in Index-Ordnung auch einen schnellen Zugriff zu einzelnen Datensaetzen. Ausserdem koennen Indexdateien beim Einfuegen, Loeschen und Aendern von Datensaetzen automatisch aktualisiert werden. REDABAS-4 indiziert auf Wunsch beliebig viele geeignete Datenfelder.

Eine Datenbank besteht bei REDABAS-4 aus einer Menge zusammengehoeriger Datenbankdateien und den dafuer aufgebauten "Hilfsdateien", z.B. den eben erwaehnten Indexdateien oder einer Katalogdatei.

### 8.1.2. Datentypen und Feldformate

#### 8.1.2.1. Datentypen

REDABAS-4 unterstuetzt die 5 unterschiedlichen Datentypen Zeichen (C), numerisch (N), logisch (L), Datum (D) und MERK (M) zum Eingeben, Speichern, Manipulieren, Anzeigen und Ausgeben von Daten und Ergebnissen.
Der Datentyp legt fest, welcher Art und Bedeutung die Daten sind, welche Manipulationen mit diesen Daten moeglich und sinnvoll sind und ob sich von vornherein eine Begrenzung des Wertevorrats ergibt. Andererseits muss REDABAS-4 bei der Manipulation von Daten die Datentypen genau kennen, um Anwendungsfehler zu erkennen und sinnlose Verknuepfungen auszuschliessen. Zu den Daten eines Datentyps gehoeren i.a. mehrere unterschiedliche konkrete Darstellungen der Daten. Je nach momentaner Bestimmung kann eine andere Darstellung benutzt sein, z.B. wird in Datumsausdruecken die fuer Berechnungen effektivste interne Zahlendarstellung gewaehlt, waehrend das Datum im Datenbankfeld eine sortierfaehige Zeichendarstellung besitzt und beim Kontakt mit dem Anwender (durch eine zwischengeschaltete typspezifische Datenkonvertierung) in der gebraeuchlichen Form tt.mm.jj oder tt.mm.jjjj angeboten wird.

Ein Datentyp besitzt in REDABAS-4:
(a) eine oder mehrere REDABAS-4 - interne Darstellungen in Datenbankfeldern, Speichervariablen und Arbeitsspeicher-Bereichen fuer die Datenmanipulation;

(Da bei der Darstellung in der Datenbankdatei und im Arbeitsspeicher unterschiedliche Gesichtspunkte Vorrang haben, unterscheiden sich diese Darstellungen auch bei gleichem Datentyp i.a. noch voneinander.)

(b) eine Darstellung als Konstante innerhalb der Befehlssprache.

Auf diese Darstellungen eines Datentyps sind die typspezifischen Operationen (s. Abschn. 8.3.3.) und Funktionen (s. Abschn. 9.2.) anwendbar.

Zusaetzlich ordnet REDABAS-4 manchem Datentyp noch standardmaessige externe Darstellungen zu, die durch Konvertierungen aus dem eigentlichen Datentyp abgeleitet werden:

(c) die Datendarstellungen in den zum Datenaustausch bestimmten Dateien;

(d) eine mehrere externe Darstellungen fuer den Kontakt zwischen REDABAS-4 und dem Anwender (z.B. fuer die Dateneingabe und -anzeige im Dialog und die Datenausgabe ueber Reports und Listen).

Fuer die standardmaessige Nutzung von REDABAS-4 sind die internen Darstellungen (a) nicht und die Datenaustausch-Darstellungen (c) selten von Bedeutung. Der Anwender kommt vorwiegend mit den externen Darstellungen (d) und den Konstanten (b) in Beruehrung.

Im folgenden werden unter Datentyp die Erlaeuterungen gegeben, die in REDABAS-4 unabhaengig von der konkreten Darstellung des Datentyps gelten. Darstellungsabhaengige Informationen sind bei Feldformaten (s. Abschn. 8.1.2.2.) und Speichervariablen (s. Abschn. 8.2.2.) zu finden.

REDABAS-4 behandelt 5 unterschiedliche Datentypen:

Typ "C" - Zeichen:

Dieser Datentyp benutzt ASCII-Zeichen ohne irgendeine spezifische Interpretation. Die Zeichen werden als Zeichenreihe betrachtet, d.h. abgesehen von ihrer Rangfolge im ASCII-Alphabet bzw. von der davon abweichend festgelegten Zeichen-Sortierfolge (siehe Abschnitt 8.3.3.2.) haben nur das Auftreten eines Zeichens oder einer Teilzeichenreihe innerhalb der Zeichenreihe und die zugehoerige relative Position eine Bedeutung.

Auf diesen Datentyp koennen die Zeichenreihen-Operationen (+, -), Teilzeichenreihen-Vergleich ($), Vergleichsoperationen und Zeichenreihen-Funktionen angewandt werden.

Zeichenreihen sind in REDABAS-4 generell auf 254 Zeichen begrenzt.

Typ "N" - numerisch:

Numerische Daten sind entweder ganze Zahlen oder Dezimalzahlen. Sie koennen positiv oder negativ sein und zu Berechnungen benutzt werden.

Auf den numerischen Datentyp koennen arithmetische Operationen (Addition, Subtraktion, Multiplikation, Division und Potenzierung, siehe Abschnitt 8.3.3.1.), Vergleichsoperationen (siehe Abschnitt 8.3.3.2.) und mathematische Funktionen (s. Abschn. 9.2.) angewandt werden. Arithmetische Operationen werden mit einer Genauigkeit von 13 bis 15 Stellen, Vergleichsoperationen (ausser mit Null) nur mit 13 Stellen ausgefuehrt.

Typ "D" - Datum:
Dieser Datentyp dient zur Aufnahme eines kalendarischen Datums (einschliesslich Jahrhundert). REDABAS-4 sichert durch Pruefung, dass ausser dem Fall "unbelegt" nur gueltige kalendarische Daten dargestellt werden.

Mit dem Datentyp D sind die Operationen (+, -) der Datumsarithmetik
- Differenzbildung zwischen zwei Datumswerten (ergibt eine Anzahl von Tagen),
- Subtraktion einer Anzahl von Tagen von einem Datum (ergibt ein neues Datum),
- Addition einer Anzahl von Tagen zu einem Datum (ergibt ein neues Datum)

und Datumsvergleiche ausfuehrbar. Ausserdem ist eine Vielzahl von Datums-Funktionen auf den Datentyp D anwendbar.

Das unbelegte Datum nimmt eine Sonderstellung ein:
- Addieren oder Subtrahieren einer Anzahl von Tagen belaesst das Datum bei unbelegt,
- die Differenz zweier unbelegter Daten ist 0,
- alle Vergleichsoperationen, an denen genau ein unbelegtes Datum beteiligt ist, liefern "falsch",
- alle Vergleichsoperationen zweier unbelegter Daten ergeben "wahr",
- Datums-Funktionen mit numerischen Ergebnissen liefern den sonst nicht auftretenden Wert 0,
- Datumsoperationen mit Zeichenreihenergebnissen liefern "Unbekannt".

Sollten diese Standardreaktionen im Anwendungsfall unerwuenscht sein, ist ein eventuell unbelegtes Datum in <ausdrD> durch einen Vergleich mittels der Bedingung

```
DTOC(<ausdrD>) = "  .  .  "
```

zunaechst auszusondern und speziell zu behandeln.

Typ "L" - logisch:

Der Datentyp L umfasst zwei moegliche Werte, die als Wahrheitswerte "wahr"/"ja" (englisch "true"/"yes") bzw. "falsch"/"nein" (englisch "false"/"no") interpretiert werden. In der Feld-Darstellung kommt als dritter Wert "unbelegt" dazu.
Moeglich sind logische Operationen (.AND., .OR., .NOT.) und die IIF()-Funktion. Andererseits liefern alle Vergleichsoperationen und viele Funktionen mit anderen Datentypen ein Ergebnis vom Typ L.

Typ "M" - MERK:

MERK ist ein spezieller Datentyp mit eingeschraenkter Verwendbarkeit innerhalb von REDABAS-4. Der Datentyp M kann z.B. nur als Feld einer Datenbankdatei auftreten, wobei Besonderheiten bei der Speicherung und Benutzung dieser Merkfelder zu beachten sind (s. Abschn. 3.6.).
Darstellbar ist beliebiger, variabel langer Text, der in REDABAS-4 nicht durch Operationen verknuepft oder Funktionen behandelt werden kann. Alle mit dem Datentyp M moeglichen Manipulationen (ausser der Verwaltung der Merkdatei und der einfachen Anzeige des Textes mit LIST, ?, REPORT FORM usw.) werden auf den Texteditor verlagert und durch ihn bestimmt.

### 8.1.2.2. Feldformate

REDABAS-4 arbeitet bei Datenbankdateien mit fester Satzstruktur und damit auch fester Satzlaenge, d.h. es wird fuer jeden Datensatz auf der Platte ein gleichgrosser, durch die Strukturdefinition festgelegter Platz reserviert. (Eine Ausnahme bildet die Speicherung der Merkfelder, wofuer je Datenbankdatei eine gesonderte Merkdatei angelegt wird.)
Die Feldattribute Datentyp, Feldlaenge und Dezimalstellenanzahl (bei Datentyp N) sind von den momentanen Feldinhalten getrennt in der Strukturdefinition der Datenbankdatei gespeichert und gelten fuer das entsprechende Feld in jedem Datensatz. Feldwerte, die nicht die definierte Feldlaenge ausnutzen, werden automatisch durch Leerstellen ergaenzt.

Jedes Feld in einer Datenbankdatei muss einen eindeutigen Namen besitzen, der mit einem Buchstaben beginnt, bis zu 10 Zeichen lang ist und ausser Buchstaben auch Ziffern und das Unterstreichungszeichen enthalten kann.

REDABAS-4 unterstuetzt 5 den Datentypen entsprechende Feldformate:

Zeichenfeld (<feldC>, Datentyp C):

Dieses Feldformat erlaubt alle druckbaren ASCII-Zeichen wie Buchstaben, Ziffern, Leerzeichen und Sonderzeichen, die mit der Tastatur erzeugt werden koennen. Andere Zeichen, die nicht druckbar sind, aber mit den Steuertasten der Tastatur auch erzeugt werden koennen, sind nicht zugelassen. Befinden sich solche Zeichen in Datenfeldern, koennen unkontrollierte Bildschirm- oder Programmeffekte auftreten.
Die Maximallaenge eines Zeichenfeldes betraegt 254 Zeichen. Sie muss auf die in den Datensaetzen der Datei fuer dieses Feld maximal moegliche Zeichenanzahl ausgelegt werden. Eingegebene Informationen werden linksbuendig dargestellt und rechts gegebenenfalls durch Leerzeichen bis zur definierten Feldlaenge ergaenzt. Der eigentliche Informationsgehalt des Feldes steht bereit, wenn die Funktion TRIM() oder RTRIM() auf den Feldinhalt angewandt wird, um die nachgestellten Leerzeichen abzuschneiden. Analog liefert LEN(<feldC>) die definierte Feldlaenge, waehrend LEN(TRIM(<feldC>)) die belegte Laenge ermittelt.

Zeichenfelder sind auch zum Speichern von Zahlen geeignet, die nur zum Identifizieren, nie aber zum Berechnen dienen (Telefonnummer, Kundennummer, Postleitzahl usw.).
Zeichenfelder koennen zum Sortieren und Indizieren benutzt und als Schluesselfelder bei TOTAL und UPDATE verwendet werden.

Beispiel: Feld NAME (Laenge 20) zum Speichern eines Namens

numerisches Feld (<feldN>, Datentyp N):

Numerische Felder speichern positive oder negative Zahlen. Bei der Felddefinition wird durch Angeben der Anzahl der Dezimalstellen bestimmt, ob das Feld ganze Zahlen ( 0 Dezimalstellen) oder Dezimalzahlen (1 bis 15 Dezimalstellen) speichern soll.

Beim Festlegen der Feldlaenge (1 bis 19) ist zu beruecksichtigen, dass fuer das Speichern vorzeichenbehafteter Zahlen ein Byte auf das Vorzeichen entfaellt. Bei Dezimalzahlen belegt der Dezimalpunkt ein Byte, und die Feldlaenge muss um mindestens zwei groesser sein als die Dezimalstellenanzahl.

Die Feldlaenge ("LF") ergibt sich aus der Anzahl der benoetigten Ziffernstellen ("LZ") wie folgt:

```
LF = LZ       fuer vorzeichenlose ganze Zahlen,
LF = LZ + 1   fuer vorzeichenbehaftete ganze Zahlen
              oder vorzeichenlose Dezimalzahlen,
LF = LZ + 2   fuer vorzeichenbehaftete Dezimalzahlen.
```

Fuer ein bestimmtes numerisches Feld der Datenbankdatei liegt die Dezimalstellenanzahl nach der Strukturdefinition ueber alle Datensaetze fest, d.h. im gleichen Feld hat der Dezimalpunkt fuer alle Datensaetze die gleiche Position. Die Standardmaske zur Dateneingabe (z.B. APPEND) enthaelt beim leeren Feld bereits den Dezimalpunkt an der richtigen Position. Deshalb muss der Dezimalpunkt nicht mit eingegeben werden, kann aber zur Justierung der eingegebenen Ziffern benutzt werden.

Eine Gleitkommadarstellung, d.h. ein Ausdruecken des Zahlenwerts durch Mantisse und Exponent, in einem Feld unterstuetzt REDABAS-4 nicht. (Als Ausweichloesung muessen Mantisse und Exponent in getrennten numerischen Feldern gespeichert werden; die ordnungsgemaesse Anzeige, das Uebertragen in eine Speichervariable zu Rechenoperationen und das Rueckfuehren des Rechenergebnisses in die beiden Felder muessen durch Programme selbst gesteuert werden.)

Bei Verwendung eines numerischen Feldes unterscheidet REDABAS-4 nicht zwischen dem Nullwert und "unbelegt" (Leerzeichen im Feld). Ausserdem kann ein unbelegtes Feld durch Manipulationen ohne gezielte Null-Eingabe zu Null werden. Muss z.B. fuer statistische Auswertungen ausdruecklich zwischen "0" und "unbelegt" unterschieden werden, ist eine Belegungsanzeige mitzufuehren und zu verwalten. Z.B. kann ein praktisch unmoeglicher Wert den Fall "unbelegt" ausdruecken oder ein zusaetzliches logisches Feld als Anzeige dienen, wobei das Anwendungsprogramm die Fallunterscheidung vornehmen muss.
Der Wertebereich , der mit einem numerischen Feld in REDABAS-4 direkt darstellbar ist, bewegt sich etwa zwischen

```
      +17        -15          -15        +17
  -10    ... -10    , 0 , +10    ... +10
```

Die Darstellungsgenauigkeit der Zahlen ist auf 15 bis 16 Ziffern begrenzt, d.h. nur die 15 hoeherwertigen Ziffern sind zuverlaessig darstellbar.

Beispiel: Speicherung einer Stueckzahl als ganze Zahl in einem numerischen Feld STK der Laenge 6 (max. 999999) und eines Preises in einem Feld PREIS der Laenge 7 mit 2 Dezimalstellen (max. 9999.99)

Datumsfeld (<feldD>, Datentyp D):

Ein Datumsfeld ist 8 Zeichen lang. Es speichert das Datum in Zeichenform, aber in sortierfaehiger Reihenfolge und generell mit Jahrhundert.
Von dieser internen Form weichen die maskengesteuerte Anzeige, die maskengesteuerte Eingabe und die Darstellung in anzeigenden Befehlen wie ? und LIST ab. Es wird dann die im taeglichen Leben gebraeuchliche Datumsform verwendet:

```
   tt.mm.jj     Bsp. 23.11.66    fuer SET CENTURY OFF
   tt.mm.jjjj   Bsp. 23.11.1966  fuer SET CENTURY ON
```

Bei der maskengesteuerten Eingabe werden die Punkte bereits angezeigt und brauchen nicht mit eingegeben zu werden. Ungueltige kalendarische Daten werden abgewiesen.
Beim Verknuepfen eines Datums zu einem Ausdruck ist zu beachten, dass das Datumsfeld unabhaengig von seiner externen Anzeige vom Datentyp D ist. Es ist streng von der Zeichenreihe tt.mm.jj bzw. tt.mm.jjjj zu unterscheiden (Datentyp C) und muss erforderlichenfalls erst durch die Funktion DTOC in den Datentyp C verwandelt werden.

```
Beispiel:      ? "faellig:", <feldD>          && richtig
               ? "faellig: "+<feldD>          && falsch
               ? "faellig: "+ DTOC(<feldD>)   && richtig
```

In der Praxis wird es oft vorkommen, dass ein Feld vom Datentyp "Datum" erst nach einer bestimmten Bearbeitung ausgefuellt wird (z.B. Erfuellungstermin). Die Besonderheiten des "unbelegten" Datums sind in Abschnitt 8.1.2.1. erlaeutert.
Datumsfelder koennen zum Sortieren und Indizieren benutzt und als Schluesselfelder bei TOTAL und UPDATE verwendet werden.

Beispiel: Speichern des Geburtsdatums in einem Feld GEB

Logisches Feld (<feldL>, Datentyp L):

Logische Felder sind ein Zeichen lang. Sie enthalten T fuer "wahr", F fuer "falsch" oder ein Leerzeichen fuer "unbelegt".
Die folgende Tabelle gibt einen Ueberblick ueber Eingabemoeglichkeiten und Anzeige:

| | "wahr" | "falsch" | "unbelegt" |
|---|---|---|---|
| maskengesteuerte Eingabe | T/t/J/j | F/f/N/n | |
| maskengesteuerte Anzeige | T | F | Leerzeichen |
| anzeigende Befehle (LIST, ?, ...) | .T. | .F. | .F. |
| Bedeutung in Bedingungen und log. Verknuepfungen | wahr | falsch | falsch |

Achtung: Bei der maskengesteuerten Eingabe gilt u.a. J/j als "wahr", waehrend Y/y nicht akzeptiert wird. In Befehlen dagegen sind die Konstanten .Y./.y. gestattet, nicht aber .J./.j. !
Logische Felder sind nicht benutzbar zum Indizieren (fuer Ausweichloesung siehe IIF()-Funktion), Sortieren und als Schluesselfelder bei TOTAL und UPDATE.

Beispiel: Ausdruecken der Tatsache, ob bezahlt oder nicht, in einem logischen Feld BEZ

Merkfeld (<feldM>, Datentyp M):

Ein Merkfeld dient zur Aufnahme variabel langer Textinformationen. Sollen Informationen gespeichert werden, die die maximale Groesse von 254 Zeichen ueberschreiten (d.h. nicht in ein Datenfeld vom Typ "Zeichen" passen), koennen sie in einem Merkfeld untergebracht werden.
Damit wird es moeglich, jedem Datensatz spezielle unformatierte Textinformationen, wie Beschreibungen, Erlaeuterungen, Hinweise und dgl. hinzuzufuegen.

In der Datenbankdatei selbst nimmt ein Merkfeld unabhaengig von seiner Groesse immer 10 Bytes ein, da dort nur ein Verweis auf eine separate Datei (Merkdatei) mit dem Dateityp .DBM gespeichert wird. Diese Merkdatei, die die Informationen der Merkfelder enthaelt, wird automatisch mit der zugehoerigen Datenbankdatei erzeugt. Der Platz wird den Merkfeldern in Portionen (Bloecken) zu 512 Bytes zugeordnet.
Pro Datensatz koennen beliebig viele der maximal 128 Felder einer Datenbankdatei als Merkfeld definiert werden.

Die Dateneingabe in ein Merkfeld erfolgt standardmaessig mit dem REDABAS-4-Texteditor, wobei die Laenge des einzelnen Merkfeldes 4096 Zeichen nicht ueberschreiten darf. Wird durch Konfigurieren ein anderer, externer Texteditor in REDABAS-4 eingebunden (siehe Abschnitt 10.1.), bestimmt dieser die maximale Feldlaenge und alle Editierungsmoeglichkeiten.

Merkfelder unterliegen wesentlichen Einschraenkungen, sie koennen nur mit bestimmten Befehlen erstellt und editiert werden. Bei der Anzeige des Inhalts durch die Befehle DISPLAY, LIST oder ? muss der Name des Merkfeldes explizit angegeben werden. Merkfelder koennen nicht fuer das Durchsuchen, Bilden von Bedingungen und Ausdruecken und nicht innerhalb von Funktionen genutzt werden. Sie sind nicht benutzbar zum Indizieren, Sortieren und als Schluesselfelder bei TOTAL und UPDATE. (weitere Informationen zu Merkfeldern und -dateien siehe Abschnitt 3.6.)

Beispiel: In einer Datei ueber Geraete werden in einem Merkfeld BEDIEN Bedienungshinweise verwaltet

### 8.1.3. REDABAS-4-Dateien

Allgemein versteht man unter einer Datei eine Sammlung von Daten, die auf einem Massenspeicher (z.B. Diskette) aufbewahrt werden. Das Betriebssystem DCP unterstuetzt diese Datenspeicherung und organisiert auf dem Speicher ein Verzeichnis, in dem bis zu 112 Dateien mit jeweils einem Namen eingetragen sind. Der Anwender kann mit dem DCP-Kommando MKDIR Unterverzeichnisse einrichten, in denen die Anzahl der Dateiverweise nicht fest begrenzt ist. Der Dateiname kann entsprechend der DCP-Konvention aus maximal acht Zeichen fuer den eigentlichen Namen, gefolgt von einem Punkt ".", und der bis zu drei Zeichen langen Erweiterung bestehen.

REDABAS-4 organisiert 15 verschiedene Dateitypen, die sich

durch ihre Bedeutung innerhalb REDABAS-4 sowie ihre innere Struktur voneinander unterscheiden. Zu jedem Dateityp benutzt REDABAS-4 einen Voreinstellwert fuer die Erweiterung. Diese Voreinstellwerte sind im allgemeinen fuer den Nutzer nicht zwingend, ihre Verwendung ist aber vorteilhaft, da dadurch bei Nennung des Dateinamens innerhalb von Befehlszeilen die explizite Angabe der Erweiterung entfallen kann. Um einen bestimmten Dateityp zu bezeichnen, wird kuenftig oft dessen Standarderweiterung als Abkuerzung verwendet.

Sollen in Ausnahmefaellen Dateien erzeugt oder verwendet werden, die nicht die von REDABAS-4 vergebenen Voreinstellwerte (Standarderweiterungen) fuer den Dateityp tragen, ist die vom Anwender gewaehlte Erweiterung in jeden Fall anzugeben.

```
Beispiel:  CREATE kunden.xyz
           INDEX ON plz TO kunden.plz
             .
             :
           USE kunden.xyz INDEX kunden.plz
```

Ein Spezialfall unter den Ausnahmen stellt auch die Vergabe ueberhaupt keiner Erweiterung fuer von REDABAS-4 zu erzeugenden oder zu benutzenden Dateien dar. In diesem Fall ist bei den jeweiligen Befehlen anstelle von <datei> (vgl. Abschn. 9.1.) - also dort, wo ein Standard fuer die Erweiterung gilt- ein Punkt hinter den bis zu 8 Zeichen langen Dateinamen zu setzen.

```
Beispiel:  Speichervariablendatei var1 (ohne Erweiterung)
         - Befehle mit <datei>
                SAVE TO var1.
                RESTORE FROM var1.

         - Befehle mit <datei.erw>
                COPY FILE var1 TO var2.var
           oder COPY FILE var1. TO var2.var
                ERASE var1
           oder ERASE var1.
```

Achtung! Die Vergabe einer vom dateitypspezifischen Standard abweichenden Erweiterung oder die Vergabe ueberhaupt keiner Erweiterung sollte Ausnahmefaellen vorbehalten sein. Die Menueangebote von Befehlen, wie z.B. ASSIST und MODIFY VIEW koennen nur dann uneingeschraenkt genutzt werden, wenn die von REDABAS-4 verwalteten Dateien die standardmaessige, den Dateityp anzeigende Erweiterung tragen.
Dateien ohne Erweiterung werden von vielen Befehlen nicht voll unterstuetzt, z.B. von CREATE oder MODIFY COMMAND.

Zum Umbenennen von nicht standardmaessig bezeichneten Dateien kann der REDABAS-4-Befehl RENAME oder das DCP-Kommando RENAME genutzt werden.

REDABAS-4 arbeitet mit folgenden Dateitypen:

| Name: | Standarderweiterung: |
|---|---|
| Datenbankdatei | .DBD |
| Katalogdatei | .CAT |
| Merkdatei | .DBM |
| Sicherungsmerkdatei | .TBK |
| Indexdatei | .IDX |
| Speichervariablendatei | .VAR |
| Programmdatei | .PRG |
| Reportdatei | .DEF |
| Etikettendatei | .ETI |
| Querydatei | .QRY |
| Screendatei | .SCR |
| Maskendatei | .MSK |
| Viewdatei | .VUE |
| Textdatei | .TXT |
| Sicherungsdatei | .BAK |

Diese Dateitypen werden im folgenden erlaeutert:

### 8.1.3.1. Datenbankdateien (.DBD)

Die Datenbankdatei ist das Kernstueck einer REDABAS-4-Datenbank. Sie enthaelt die Strukturbeschreibung der Datensaetze entsprechend der vom Nutzer vorgenommenen Definition im CREATE-, MODIFY STRUCTURE- oder CREATE SCREEN-Befehl. Die Strukturbeschreibung kann maximal 128 Eintraege umfassen. Jeder Eintrag beschreibt ein Datenfeld und enthaelt die Angaben

- Feldname
- Datentyp
- Laenge des Datenfeldes
- Anzahl der Dezimalstellen.

Nach der Strukturbeschreibung sind in der Datenbankdatei die eigentlichen Datensaetze angeordnet. Ein Datensatz kann maximal 4000 Bytes lang sein. Die Anzahl der Datensaetze darf 1 Milliarde nicht ueberschreiten, eine Datei kann sich nur ueber einen Datentraeger erstrecken.

Datenbankdateien lassen sich aus bereits bestehenden Datenbankdateien durch Aendern der Dateistruktur (MODIFY STRUCTURE), Sortieren (SORT), Kopieren (COPY), Verdichten (TOTAL) oder Verknuepfen (JOIN) gewinnen.
Datenbankdateien sollten nicht mit Texteditoren oder aehnlichen Programmen bearbeitet werden!

### 8.1.3.2. Katalogdateien (.CAT)

Katalogdateien sind spezielle Datenbankdateien mit festgelegter Struktur (s. Abschn. 6.3.) und dem Dateityp .CAT . Sie dienen zur besseren Gliederung und Uebersicht der Dateien eines Datenbanksystems. Zur Katalogfuehrung dienen eine Hauptkatalogdatei CATALOG.CAT im aktuellen Verzeichnis des aktuellen Laufwerks und Katalogdateien, deren Namen zur Entstehungszeit frei waehlbar waren, jedoch mit dem zwingenden Dateityp .CAT . In der Hauptkatalogdatei werden Katalogdateien, in der Katalogdatei

Datenbank-, Index-, Report-, Etiketten-, Query-, Screen-, Masken- und Viewdateien gefuehrt.

Die Katalogfuehrung wird mit dem Befehl SET CATALOG TO [<katalogdatei>/?] aktiviert oder entaktiviert. REDABAS-4 eroeffnet die im Befehl spezifizierte oder infolge einer Katalog-Frage-Klausel ausgewaehlte Katalogdatei im Arbeitsbereich 10 (vgl. hierzu Kap. 6. Die Benutzung von Katalogdateien).

### 8.1.3.3. Merkdateien (.DBM)

Merkdateien sind einer Datenbankdatei zugeordnete Hilfsdateien, die bei der Definition von Merkfeldern automatisch erzeugt werden. Alle Merkfelder einer Datenbankdatei werden in derselben Merkdatei gespeichert.

Datenbank- und Merkdatei gehoeren zusammen. Geht die Merkdatei verloren, ist die zugehoerige Datenbankdatei nicht benutzbar.
Ein Satz einer Datenbankdatei kann bis zu 128 Merkfelder enthalten, wobei in der Datenbankdatei selbst nur 10 Stellen pro Merkfeld belegt sind. In der Merkdatei nimmt jedes Merkfeld pro Satz mindestens 512 Bytes ein, wenn Daten eingegeben werden. Maximal koennen 4096 Zeichen (Standardmaessig, vgl. Abschn. 10.1.3.) gespeichert werden. Die Beibehaltung des Dateityps .DBM ist erforderlich (vgl. hierzu Abschn. 3.6. Die Arbeit mit Merkfeldern).

### 8.1.3.4. Sicherungsmerkdateien (.TBK)

Wird der Befehl MODIFY STRUCTURE fuer eine Datenbankdatei mit Merkfeldern benutzt, wobei die Anzahl oder die Namen der Merkfelder modifiziert werden, sichert REDABAS-4 die vorhandene alte Version der Merkdatei und vergibt dafuer den Dateityp .TBK (vgl. Abschn. 3.6.). Der Nutzer hat keinen Einfluss auf die Vergabe eines anderen Dateityps. Die modifizierte Merkdatei erhaelt den Dateityp .DBM .

### 8.1.3.5. Indexdateien (.IDX)

Indexdateien werden durch den INDEX-Befehl erzeugt und enthalten neben einer Eintragung ueber den Schluesselausdruck zu jedem Datensatz in der Datenbankdatei den aktuellen Wert des Schluesselausdrucks und einen Adressverweis zum zugehoerigen Datensatz.
Die Indizierung bewirkt die geordnete Bereitstellung der Datenbankdatei nach einem Schluessel, ohne dass eine Umordnung der Datenbankdatei stattfindet. Die Indexdatei beschleunigt den Zugriff auf einzelne Datensaetze vor allem bei groesseren Datenbankdateien.
Die Aktivierung des Indexes erfolgt mit der INDEX-Klausel im USE-Befehl oder mit SET INDEX TO (vgl. hierzu Abschn. 3.4.2. Indizieren).

### 8.1.3.6. Speichervariablendateien (.VAR)

REDABAS-4 legt eine Speichervariablendatei an, wenn der Nutzer mit dem Befehl SAVE die in einer REDABAS-4-Sitzung erzeugten Speichervariablen fuer die Nutzung in weiteren Programmlauefen permanent speichern will. Die Speichervariablendatei kann maximal 256 Speichervariablen aufnehmen. Eine Speichervariable kann Konstanten, Feldinhalte oder Ergebnisse von Berechnungen enthalten. Es gibt 4 Arten von Speichervariablen: Zeichen, Numerisch, Logisch, Datum.
Durch den Befehl RESTORE wird die Speichervariablendatei zur Wiederverwendung in den Arbeitsspeicher geladen (vgl. hierzu Abschnitt 8.2. Speichervariablen).

### 8.1.3.7. Programmdateien (.PRG)

Hauefig benutzte Befehlsfolgen kann der Nutzer z.B. mit dem Befehl MODIFY COMMAND in einer Programmdatei speichern und mit dem DO-Befehl jederzeit zur Ausfuehrung bringen.

Solche Programmdateien koennen komplette Programme enthalten. Die Schachtelung von Programmdateien ist moeglich, d.h. in einer Programmdatei kann durch den DO-Befehl eine weitere Programmdatei aufgerufen werden. Die moegliche Schachtelungstiefe ist durch die maximale Anzahl von 15 gleichzeitig eroeffneten Dateien begrenzt, wobei jeder DO-Befehl das Eroeffnen der entsprechenden Programmdatei bewirkt. Ausserdem sind in der Zahl 15 auch die anderen momentan eroeffneten Dateien zu beruecksichtigen (z.B. Datenbankdateien, Indexdateien, Reportdateien).

Die Ausfuehrung einer Programmdatei wird beendet, wenn entweder das Ende der Datei erreicht ist oder der Befehl RETURN abgearbeitet wird. Nach der Ausfuehrung steht der durch die Programmdatei belegte Platz im Arbeitsspeicher anderen Programmdateien zur Verfuegung.
Programmdateien sind ASCII-Dateien. Sie koennen entweder mit MODIFY COMMAND oder mit Textverarbeitungssystemen erstellt und geaendert werden.
Im Normalfall enthaelt eine Programmdatei nur ein Programm.
Ein Sonderfall einer Programmdatei ist die P r o z e d u r - datei, die eine Sammlung von bis zu 32 als Prozedur gekennzeichneten Programmen enthalten kann (vgl. hierzu Abschnitt 4.4.). Bei Benutzung von Prozeduren koennen bis zu 20 DO-Aufrufe verschachtelt werden.

### 8.1.3.8. Reportdateien (.DEF)

Solch eine Datei erzeugen die Befehle CREATE REPORT bzw. MODIFY REPORT automatisch, wenn der Dialog zur Listengestaltung gefuehrt wird. Die Reportdatei enthaelt alle vom Nutzer waehrend des REPORT-Dialogs eingegebenen Angaben zur Steuerung des Ausgabeformats der gewuenschten Ergebnisliste.
Mit REPORT FORM wird eine Reportdatei eroeffnet, und entsprechend den in dieser Datei enthaltenen Anweisungen wird ein Report aus den Daten der aktuellen Datenbankdatei erstellt.

### 8.1.3.9. Etikettendateien (.ETI)

Etikettendateien werden mit den Befehlen CREATE LABEL bzw. MODIFY LABEL erzeugt bzw. veraendert. Sie beinhalten die Daten fuer das Ausgabeformat der Etiketten (z.B. Adressaufkleber).
Der Befehl LABEL FORM eroeffnet die Etikettendatei und sorgt fuer die Formatspezifikation der Etiketten, deren Inhalt aus der aktuellen Datenbankdatei entnommen wird.

### 8.1.3.10. Querydateien (.QRY)

Querydateien werden durch Aufruf der Befehle CREATE QUERY bzw. MODIFY QUERY erstellt bzw. modifiziert. Sie enthalten eine komplexe Filterbedingung, die sich aus der Verknuepfung von bis zu 7 einfachen Filterbedingungen zusammensetzt. Eine Querydatei kann einer Datenbank- oder Viewdatei zugeordnet werden. Es werden dann nach Befehlen, deren Syntax <bereich> aufweist, nur solche Datensaetze der Datenbankdatei oder der in einer Viewdatei verbundenen Datenbankdateien bereitgestellt, die die komplexe Filterbedingung erfuellen.
Der Befehl SET FILTER TO [FILE <querydatei>/?] aktiviert die im Befehl spezifizierte oder die durch die Katalog-Frage-Klausel ausgewaehlte Querydatei oder entaktiviert sie (vgl. hierzu Abschn. 3.9.).

### 8.1.3.11. Screendateien (.SCR)

Screendateien enthalten gespeicherte Bildschirmmasken, die der mit CREATE SCREEN bzw. MODIFY SCREEN aufgerufene Maskengenerator zur Erstellung bzw. Aenderung von Maskendateien verwendet. Sie sind ein Hilfsmittel zur Gestaltung individueller Ein-/Ausgabemasken am Bildschirm (vgl. hierzu Abschnitte 3.10. und 8.1.3.12.).

### 8.1.3.12. Maskendateien (.MSK)

Maskendateien enthalten als Ergebnis eines Maskengeneratorlaufs ausschliesslich @-Befehle und im Falle mehrseitiger Ein-/Ausgabemasken READ-Befehle. Ein @-Befehl bewirkt die Einstellung auf eine bestimmte Cursor- oder Druckposition und ermoeglicht in Verbindung mit Parametern und eingefuegten Befehlen eine programmierte Dateneingabe/-ausgabe. Ein READ-Befehl loest die Aktivierung der vorhergehenden @ <koordinaten> GET-Befehle aus, loescht den Bildschirm und stellt die @-Befehle der neuen Bildschirmseite zur Verfuegung. Der Nutzer kann mit <PgUp> bzw. <PgDn> zwischen den Bildschirmseiten vor- bzw. zurueckblaettern. Voraussetzung dafuer ist, dass der SET FORMAT TO- Befehl vorher gegeben wurde.

Durch den Befehl SET FORMAT TO <maskendatei>/? wird diese Datei im Arbeitsspeicher bereitgestellt und durch nachfolgende READ-, APPEND-, CHANGE-, EDIT- oder INSERT-Befehle darauf zugegriffen. Bei Verwendung einer Maskendatei sollte keine Druck-

ausgabe mit SET DEVICE TO PRINT erfolgen. Es wuerden nur die @ <koordinaten> SAY-Befehle gedruckt. Eine Maskendatei wird im aktuellen Arbeitsbereich mit SET FORMAT TO geschlossen; dagegen schliesst CLOSE FORMAT alle Maskendateien.

Maskendateien lassen sich in REDABAS-4 auf elegante Weise mit dem Maskengenerator erstellen und aendern. Dabei muss immer eine Screendatei angesprochen werden (s. Abschn. 8.1.3.11.). Die Maskendatei erhaelt den Dateinamen der Screendatei mit dem Dateityp .MSK .

Maskendateien koennen auch mit dem Befehl MODIFY COMMAND und der Angabe des Dateinamens einschliesslich des Dateityps erstellt bzw. modifiziert werden. Ebenfalls lassen sich Textverarbeitungsprogramme dazu verwenden. Nach solch einem Eingriff in eine Maskendatei durch ein dem Maskengenerator fremdes Programm ist keine ordnungsgemaesse Aktualisierung mit dem Maskengenerator mehr moeglich. Maskendateien koennen in Programmdateien kopiert werden oder der Befehl SET FORMAT TO <maskendatei> in einer Programmdatei sorgt fuer die Aktivierung der Maskendatei (vgl. Abschn. 4.5.).

### 8.1.3.13. Viewdateien (.VUE)

Viewdateien werden durch Aufruf der Befehle CREATE VIEW bzw. MODIFY VIEW erstellt bzw. modifiziert. Sie dienen hauptsaechlich zur Speicherung bzw. Aktivierung von maximal 9 Verbindungen (SET RELATION-Befehle) zwischen Datenbankdateien. Darueber hinaus enthalten Sie folgende Informationen:

- Namen der zu oeffnenden Datenbankdateien (maximal 10 ohne geoeffneter Katalogdatei, andernfalls 9) und Indexdateien (maximal 7 pro Datenbankdatei) mit zugehoerigen Nummern der Arbeitsbereiche (jedoch maximal nur 13 Dateien als Datenbank-, Index-, Katalog- und Maskendatei),

- die Nummern des auszuwaehlenden Arbeitsbereichs,

- eine zu aktivierende Feldfolge (wahlweise),

- eine zu oeffnende Maskendatei (wahlweise),

- eine zu aktivierende Filterbedingung (wahlweise).

Der Befehl SET VIEW TO <viewdatei>/? eroeffnet die im Befehl spezifizierte oder die durch die Katalog-Frage-Klausel ausgewaehlte Viewdatei. Daraufhin werden die in der Viewdatei hinterlegten REDABAS-4- Ressourcen, die zuvor beschrieben wurden, aktiviert. Die Viewdatei wird automatisch geschlossen. Zur Entaktivierung der durch die Viewdatei aktivierten Ressourcen muss der Befehl CLOSE DATABASE (bei geoeffneter Katalogdatei) bzw. CLOSE ALL (bei geoeffneter Datenbankdatei im Arbeitsbereich 10) gegeben werden.

### 8.1.3.14. Textdateien (.TXT)

Textdateien bestehen aus Saetzen, die sich aus Zeichen des ASCII-Zeichensatzes zusammensetzen. Jeder Satz ist entsprechend der DCP-Konvention mit Wagenruecklauf und Zeilenvorschub abgeschlossen.

Die Verwendung von Textdateien bildet die Schnittstelle zwischen REDABAS-4 und anderen Programmen.

Textdateien koennen als Eingabedateien im APPEND FROM-Befehl im Zusammenhang mit der Angabe "SDF" oder "DELIMITED" benutzt werden.
Der COPY TO-Befehl mit der Angabe "SDF" oder "DELIMITED" erzeugt aus REDABAS-4-Datenbankdateien Ausgabedateien im ASCII-Format, die durch andere Programme weiterverarbeitet werden koennen.

Textdateien entstehen auf Anforderung im Maskengeneratorlauf. Sie dokumentieren die entworfene Ein-/Ausgabemaske.
Textdateien koennen auch das Ergebnis von SET ALTERNATE-Befehlen sein und stellen dann ein Protokoll von Bildschirmeingaben und -ausgaben dar.

### 8.1.3.15. Sicherungsdateien (.BAK)

Wird der Befehl MODIFY COMMAND benutzt, um eine Programmdatei, Maskendatei, Textdatei oder eine andere Standard-ASCII-Datei zu aendern, sichert REDABAS-4 die vorhandene alte Version und vergibt dafuer den Dateityp .BAK.
Der Nutzer kann hier keinen eigenen Dateityp vergeben.

## 8.2. Speichervariablen

### 8.2.1. Ueberblick

Speichervariablen dienen der Zwischenspeicherung von Daten ausserhalb der Datenbankdatei-Struktur:

- Zwischenspeichern von Tastatureingaben, Datenfeldinhalten, Ausdruckswerten und (Zwischen-)Ergebnissen zur weiteren Verwendung, z.B. auch zum Modifizieren von Befehlen durch die Makro-Ersetzung
- Parameteruebergabe an Programme, zwischen Programmstufen und Rueckmeldung von Ergebnissen aus Programmen
- temporaeres Aufbewahren von Zwischenwerten bei der interaktiven Arbeit

Die Speichervariablen haben somit innerhalb der Programme ("Programmebene(n)") ebenso Bedeutung wie bei der interaktiven Arbeit auf Befehlsebene und zum Datenaustausch (Parameter/Ergebnisse) zwischen diesen Ebenen. Sie sind eine wesentliche Voraussetzung fuer das Erstellen komplexer hierarchischer Programme.

Eine Speichervariable ist durch einen Namen, einen Datentyp, einen Wert, den Existenzbereich und ihre Sichtbarkeit gekennzeichnet. Wird eine Speichervariable angelegt, werden diese Charakteristika ("Attribute") im Arbeitsspeicher abgelegt.
Die folgende Tabelle zeigt die Attribute von Speichervariablen und die wesentlichsten REDABAS-4-Befehle, die diese Attribute beeinflussen.

| Attribut | Auspraegungen | beeinflussende Befehle |
|---|---|---|
| Name (Identifikation) | bis zu 10 Zeichen (wie Datenfeld-Namen) | Zuweisungen, PUBLIC, PARAMETERS |
| Existenz | existent oder nicht existent | Zuweisungen, PUBLIC, PARAMETERS, explizite Freigaben |
| Sichtbarkeit (Ansprechbarkeit) | sichtbar (ansprechbar) unsichtbar (verborgen) | PRIVATE, DO...WITH... |
| Existenzbereich (Lebensdauer) | global (publik) lokal (privat) | PUBLIC, PRIVATE, PARAMETERS |
| Datentyp | C, D, N, L | |
| Wert | typabhaengig variabel | Zuweisungen |
| weitere typabhaengige Attribute (z.B. Laenge fuer C, Stelligkeit fuer N) | | |

Solange REDABAS-4 aktiv ist, dient ein gesonderter Bereich im internen Arbeitsspeicher des Computers zur Aufnahme der Speichervariablen. Maximal 256 Speichervariable koennen gleichzeitig existieren und bis zu 6000 Bytes Gesamtplatz belegen. Die Bereichsgroesse kann aber im Unterschied zur maximalen Anzahl vor dem REDABAS-4-Start ueber eine Konfigurierungsangabe (MVARSIZ in der Datei REDAB.CF, siehe Abschnitt 10.1.) auf Wunsch anders gewaehlt werden.

Die Befehle DISPLAY/LIST MEMORY zeigen die wesentlichsten Attribute der momentan existierenden Speichervariablen an, ausserdem deren Gesamtanzahl, den bisher belegten und den im Speicherbereich verbleibenden freien Platz.
Mit der TYPE-Funktion kann ueberprueft werden, ob eine Speichervariable existent und ansprechbar ist und welchen Datentyp sie besitzt.

### 8.2.2. Datentypen von Speichervariablen

REDABAS-4 unterstuetzt bei Speichervariablen vier Datentypen, wobei sich die Verschluesselung der Daten i.a. von der Darstellung in den Datenbankfeldern unterscheidet:

Zeichen (<speichvarC>, Datentyp C):
Im Unterschied zu Datenbankfeldern duerfen in Speichervariablen mittels der CHR()-Funktion auch beliebige nicht druckbare Zeichen (ausser CHR(0)) eingebracht werden, z.B. pseudografische Zeichen, Steuerzeichen und ESC-Folgen fuer die Druckersteuerung.
Numerische Daten koennen mit der STR()-, logische mit der IIF()- und Datumsausdruecke mit DTOC()-Funktion in Zeichenreihen verwandelt werden, um in <speichvarC> eintragbar zu sein.
Dieser Speichervariablentyp dient u.a. zur Makro-Ersetzung (siehe Abschnitt 9.4.).
Der von der Speichervariablen eingenommene Speicherplatz ist um 2 Bytes groesser als die Laenge (maximal 254) der eingetragenen Zeichenreihe.

numerisch (<speichvarN>, Datentyp N):
Die Zahl ist in binaerer Gleitkommadarstellung (Laenge 8 Bytes) gespeichert. Wertevorrat und Genauigkeit sind bei den arithmetischen Operationen (Abschnitt 8.3.3.1.) erlaeutert.
Die Speichervariable belegt 9 Bytes im Arbeitsspeicher.

Datum (<speichvarD>, Datentyp D):
Das Datum ist einschliesslich Jahrhundert in 8 Bytes rechenfaehig verschluesselt. Die Speichervariable belegt 9 Bytes.

logisch (<speichvarL>, Datentyp L):
Der Wahrheitswert "wahr" oder "falsch" ist in einem Byte verschluesselt, der von der Speichervariablen eingenommene Platz betraegt jedoch 2 Bytes.

Bei der Nutzung logischer Speichervariablen ist zu beachten, dass eine Variable <speichvarL> bereits einen elementaren logischen Ausdruck <ausdrL> und damit eine komplette Bedingung darstellt. Statt z.B.

IF <speichvarL> = .F.
ist
IF .NOT. <speichvarL>
zu formulieren. Das gilt aber nicht nur fuer Speichervariable (<speichvarL>), sondern analog fuer Felder (<feldL>) und Funktionen mit logischem Ergebnis wie z.B. EOF().

### 8.2.3. Speichervariablen-Verwaltung

REDABAS-4 erlaubt eine "Kellerung" von Speichervariablen, d.h. eine Speichervariable kann voruebergehend unsichtbar (nicht ansprechbar) existieren, ihre Sichtbarkeit kann gegenueber ihrem Existenzbereich eingeschraenkt sein (siehe Abschnitt 8.2.5.).

Ausserdem wird das Prinzip der Speichervariablen-Verwaltung in REDABAS-4 wesentlich durch die interpretative Abarbeitung der Befehle bestimmt.
REDABAS-4 benutzt (mit Ausnahme der Befehle PUBLIC und PARAMETERS) keine explizite Speichervariablen-Vereinbarung. Jede Wertzuweisung kann implizit eine Speichervariablen-Definition beinhalten:

- existiert zum angefuehrten Variablennamen bereits eine sichtbare Speichervariable, wird diese neu belegt;

- existiert keine Speichervariable dieses Namens oder ist sie unsichtbar, dann wird eine neue Variable unter diesem Namen eingerichtet und belegt.

Obwohl die REDABAS-4-Befehlssprache beim Bilden von Ausdruecken (Funktionen, Operationen) strenge Datentyp-Anforderungen stellt (siehe Abschnitt 8.3.3.), traegt eine Speichervariable keinen ueber ihren gesamten Existenzbereich festgelegten Datentyp. Der aktuelle Datentyp und weitere typabhaengige Attribute sind durch die jeweils letzte vorangegangene Zuweisung gesetzt worden und koennen bei jeder neuen Zuweisung veraendert werden. (Im Interesse der Uebersichtlichkeit sollten Datentypwechsel natuerlich vermieden werden.)

Damit unterscheidet sich die Zuweisung zu Speichervariablen von der Aenderung von Datenbankfeldern: Aenderungen eines Feldinhalts (z.B. durch REPLACE oder EDIT) betreffen nur den Wert selbst und lassen alle anderen Attribute unberuehrt; die Attribute sind losgeloest vom einzelnen Feldwert in der Strukturbeschreibung gespeichert und gelten fuer alle Datensaetze.

Bild 100 zeigt in einer Uebersicht die zur Speichervariablen-Verwaltung in REDABAS-4 dienenden Befehle und ihre Aufgabe.

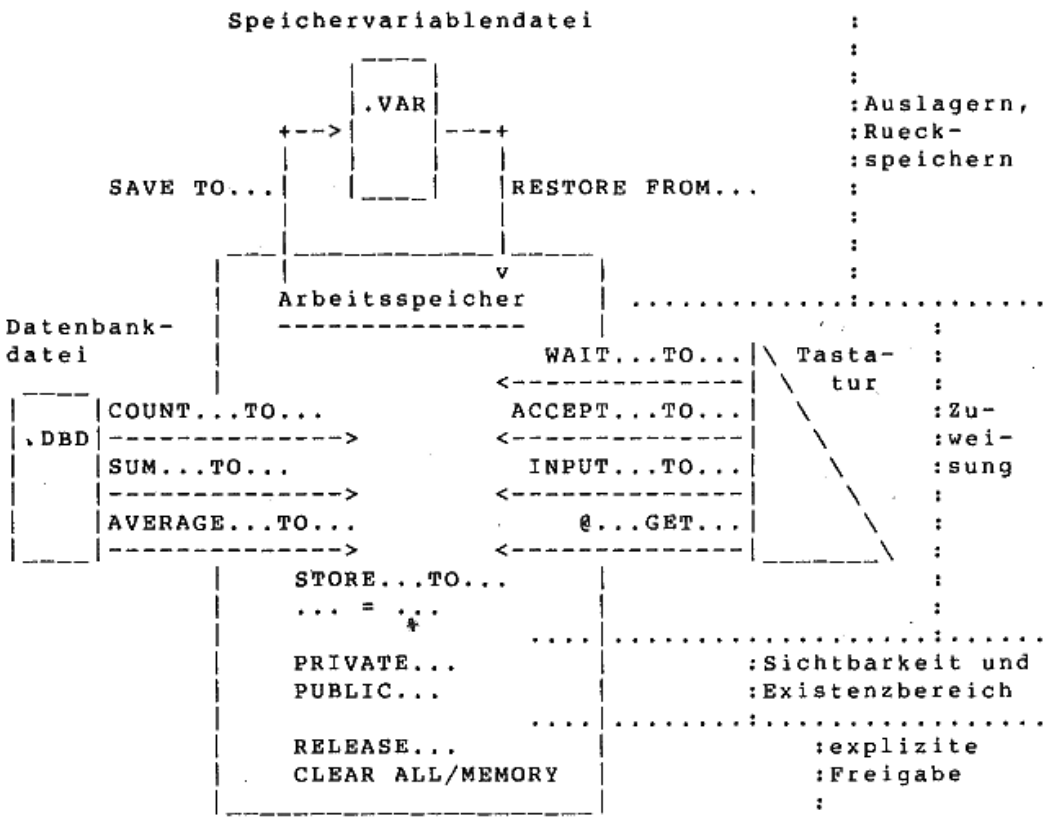



Bild 100 Uebersicht zur Speichervariablen-Verwaltung

### 8.2.4. Zuweisungen

Die folgende Tabelle stellt die Befehle zur Zuweisung zusammen (Anzahl der betroffenen Speichervariablen: 1 - fuer <speichvar>, n - fuer <speichvarfolge>). Die ausfuehrliche Syntax und Einzelheiten zu den Befehlen sind Abschnitt 9.3.3. zu entnehmen.

| Befehl | Anzahl der betroffenen Speichervariablen | Bedeutung |
|---|---|---|
| STORE ... TO...<br>... = ... | n<br>1 | explizite Zuweisung<br>" " |
| COUNT ... TO...<br>SUM ... TO...<br>AVERAGE...TO... | 1<br>n<br>n | Zaehlung von Datensaetzen<br>Summation ueber Datensaetze<br>Mittelwertbildung ueber Datensaetze |
| WAIT ... TO...<br>ACCEPT... TO...<br>INPUT ... TO... | 1<br>1<br>1 | Warten auf ein einzelnes Zeichen<br>Erwarten Zeichenreihe<br>Erwarten Ausdruck |
| @ ... GET ... | 1 | Editieren Inhalt einer existierenden Speichervariablen |
| DO... WITH ...<br>mit<br>PARAMETERS ... | n | Parameteruebergabe an ein [Unter-]Programm (als dessen lokale Speichervariable) |

Bei der expliziten Wertzuweisung kann auf den urspruenglichen Inhalt der betroffenen Speichervariablen Bezug genommen werden, z.B.

```
STORE s1+s2+s3 TO s1,s2,s3
s4 = s4+1.23
s5 = TRIM(feld)+" "+DTOC(DATE())+s5
```

Die nachfolgende Tabelle zeigt Beispiele zu den Befehlen WAIT, ACCEPT, INPUT. Der auf das Befehlswort folgende Zeichenausdruck <ausdr0> sollte mit ": " oder ">" enden, z.B.

```
INPUT "Temperatur: " TO stemp
```

da der Cursor bei der Anzeige unmittelbar hinter dem letzten Zeichen dieses Ausdrucks verbleibt.
Nach INPUT kann der Datentyp der Konsoleingabe mittels TYPE()-Funktion abgefragt werden, z.B.

```
IF TYPE("sv")="D"
```

Die Befehle STORE, =, COUNT, SUM, AVERAGE, WAIT, ACCEPT und INPUT koennen neue Speichervariable erstellen oder bestehende ueberschreiben.
PARAMETERS legt in jedem Falle neue lokale Variable an.
Der Befehl @...GET kann keine neue Speichervariable erstellen. Dafuer erlaubt er das Editieren des Inhalts einer existierenden, sichtbaren Speichervariablen, wobei neben einer impliziten Definition auch Laengen- und Datentypaenderungen ausgeschlossen sind. Diese Ausnahme unter den Zuweisungen in REDABAS-4 begruendet sich damit, dass @...GET auch zur Aenderung eines Feldinhalts in einer Datenbankdatei dient.

| <befehlswort> [<ausdrC>] TO sv | | | | |
|---|---|---|---|---|
| | | Ergebnis in sv | | |
| Befehl | Eingabe | Datentyp | Inhalt/Wert | LEN(sv) |
| WAIT | <ET> | C | - | 0 1) |
| ACCEPT | <ET> | C | - | 0 1) |
| INPUT | ""<ET> | C | - | 0 1) |
| WAIT | a | C | a | 1 |
| ACCEPT | a<ET> | C | a | 1 |
| INPUT | "a"<ET> | C | a | 1 |
| ACCEPT | abc<ET> | C | abc | 3 |
| INPUT | "abc"<ET> | C | abc | 3 |
| INPUT | .Y.<ET> | L | .T. | - 2) |
| | 127<ET> | N | 127 | - |
| | CTOD("1.2.55")<ET> | D | 01.02.55 | - |
| | CTOD("1.2.55")+30<ET> | D | 03.03.55 | - |

<ET>: Eingabetaste
1) Abfrage: IF LEN(sv)=0 oder IF ""=sv , nicht IF sv=""
2) Abfrage: IF sv Anzeige: .T. (nicht .J. oder .Y.)

### 8.2.5. Existenzbereich und Sichtbarkeit

REDABAS-4 gestattet zwei Existenzbereiche (auch "Gueltigkeitsbereiche" oder "Speicherklassen") fuer Speichervariablen:

- global (publik, permanent)
  Die Speichervariable existiert bis zum Beenden von REDABAS-4 durch QUIT oder bis zur vorherigen expliziten Freigabe.

- lokal (privat, temporaer)
  Solche Speichervariable sind an die Abarbeitung einer Programmstufe gebunden und werden bei Verlassen des [Unter-]Programms (Ende des Durchlaufs) automatisch freigegeben, sofern nicht vorher eine explizite Freigabe angefordert wurde.

Alle auf der Befehlsebene definierten Speichervariablen sind global. Die auf der Programmebene, also waehrend der Abarbeitung eines REDABAS-4-Befehlsprogramms bzw. einer Prozedur entstehenden Speichervariablen gelten standardmaessig lokal zu diesem [Unter-]Programm bzw. dieser Prozedur. Sie koennen aber auf Wunsch mit dem Befehl

PUBLIC <speichvarfolge>

als global existierend vereinbart werden. Dieser Befehl muss vor der ersten Wertzuweisung fuer die benannten Speichervariablen stehen. Globale Speichervariable sind nur noetig, wenn die Rueckmeldung von Programm-Ergebnissen bis auf die Befehlsebene, z.B. fuer ihre Nutzung in spaeter aufzurufenden Befehlen oder Programmen, beabsichtigt ist.

Standardmaessig sind alle existierenden Speichervariablen auch "sichtbar", d.h. auf sie kann zugegriffen werden, eine Zuweisung ist dann eine Bezugnahme auf die existierende Speichervariable, nicht eine Definition einer neuen Speichervariablen. Speziell in [Unter-]Programmen erstreckt sich die Sichtbarkeit auf

- alle globalen Speichervariablen,
- die eigenen lokalen Speichervariablen des [Unter-]Programms,
- die lokalen Speichervariablen aller uebergeordneten Programmstufen.

Dieser Standardfall hat den Vorteil des uneingeschraenkten Zugriffs und den Nachteil, dass Namenskonflikte der Speichervariablen ueber alle Programmstufen und die Befehlsebene durch Sorgfalt vermieden werden muessen. Das beabsichtigte Einfuehren einer neuen Variablen durch eine Zuweisung fuehrt sonst zur Verfaelschung einer bestehenden gleichnamigen Speichervariablen. Die noetige Koordinierung von Programm und Umgebung, die ueber die blossen Anschlussbedingungen hinaus geht, kann das Erstellen komplexer Programme behindern.

Da REDABAS-4 keine explizite Definition der Speichervariablen vorsieht, sind Namenskonflikte nicht durch die bei vielen konventionellen, zum Compilieren bestimmten Programmiersprachen uebliche automatische Kellerung der Variablen loesbar. Deshalb gestattet REDABAS-4 die explizite Kellerung durch den Befehl

```
PRIVATE <speichvarfolge>
PRIVATE ALL [LIKE/EXCEPT <muster>]
```

Alle in PRIVATE ausgewaehlten sichtbaren aeusseren (globalen oder lokalen) Speichervariablen werden temporaer verborgen, d.h. bis zum Rueckkehren aus dem [Unter-]Programm, in dem der PRIVATE-Befehl steht. Das Verbergen ("Kellern") geschieht ohne Aenderung des Wertes und anderer Attribute ausser Sichtbarkeit.

Damit ein Programm seine Umgebung nicht stoert, soll es alle fuer eigene lokale Speichervariable vorgesehenen Namen in PRIVATE benennen. Ob diese Namen in der Umgebung wirklich auftreten bzw. die zugehoerigen Speichervariablen bereits verborgen sind, ist bedeutungslos.

Beispiel:
Wenn von der Befehlsebene aus ein dreistufiges Programm aufgerufen wird (Bild 101), kann derselbe Variablenname s0 bis zu vier verschiedene Speichervariable bezeichnen:

eine globale verborgene
zwei lokale verborgene
eine lokale sichtbare (niedrigste Programmebene)

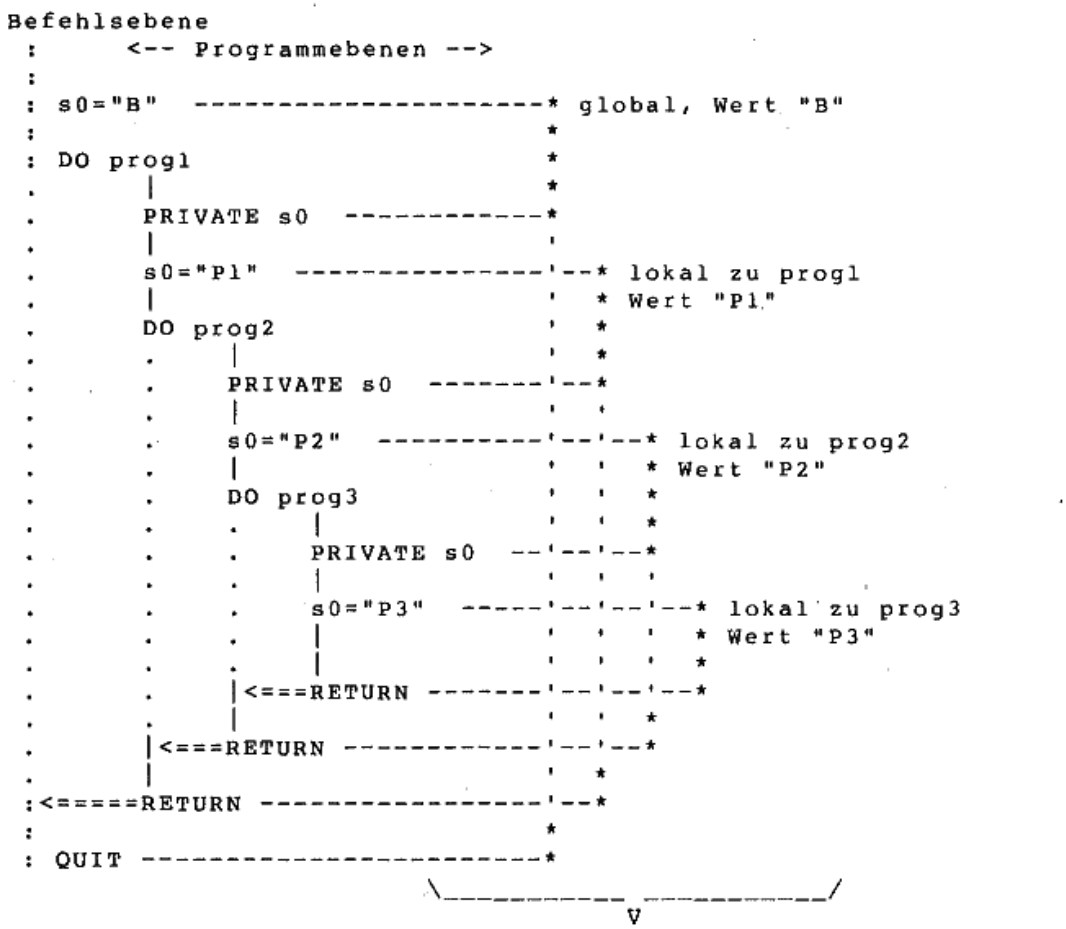



Bild 101 Beispiel zur Kellerung der Speichervariablen

Im Beispiel wurde gezeigt, dass lokale Speichervariable aufhoeren zu existieren, wenn das [Unter-]Programmm verlassen wird, in dem sie angelegt wurden, und dass globale bis zum Ende der REDABAS-4 - Sitzung gelten. Die Existenz von Speichervariablen kann aber durch explizite Freigabe vorzeitig beendet werden. Das geschieht selektiv durch RELEASE oder generell durch CLEAR MEMORY bzw. CLEAR ALL (siehe Abschnitt 9.3.3.). CLEAR MEMORY/ALL loescht alle Speichervariablen ungeachtet ihrer Attribute.
Der RELEASE-Befehl ignoriert verborgene Speichervariablen. Wird eine Speichervariablenfolge im Befehl angegeben, entfernt REDABAS-4 die aufgefuehrte Variablen unabhaengig davon, ob sie global gelten oder lokal zu irgendeiner Programmstufe. RELEASE ALL ... innerhalb eines Programms bezieht sich nur auf die "eigenen" lokalen Speichervariablen dieses Programms, waehrend lokale der uebergeordneten Programme und globale unberuehrt bleiben.

REDABAS-4 benutzt den Befehl SAVE zum Kopieren der Speichervariablen in eine Speichervariablendatei (.VAR) und RESTORE zum Rueckladen in den Arbeitsspeicher.

SAVE lagert nur die momentan sichtbaren Variablen aus und ignoriert verborgen existierende. Im Arbeitsspeicher ergeben sich dabei keine Veraenderungen.
RESTORE ohne ADDITIVE fuehrt zunaechst ein CLEAR MEMORY aus und erteilt den durch Rueckspeicherung erzeugten Speichervariablen das Attribut global, wenn der Befehl auf der interaktiven Ebene ausgefuehrt wird, sonst lokal zum jeweiligen Programm. Damit wird das neue Attribut durch die Situation zum RESTORE- und nicht zum SAVE-Zeitpunkt bestimmt.
Wird ADDITIVE angegeben, bleiben die im Arbeitsspeicher vorhandenen Speichervariablen erhalten. Existiert zum Namen einer einzulagernden bereits eine sichtbare Speichervariable im Arbeitsspeicher, findet eine Zuweisung statt (d.h. Wert, Datentyp und weitere typabhaengige Attribute koennen sich aendern, waehrend der Existenzbereich unberuehrt bleibt). Andernfalls (der Name der einzulagernden Variablen existiert noch nicht sichtbar) erfolgt die Behandlung unabhaengig von ADDITIVE.

Die Variablendatei enthaelt Wert und Attribute in verschluesselter Form, ist also zum Datenaustausch mit externen Programmen nicht vorgesehen.

### 8.2.6. Namenskonflikte zwischen Speichervariablen und Feldern

Die Namen von Speichervariablen werden wie die Namen von Feldern aus maximal 10 Zeichen (Buchstaben, Ziffern, Unterstreichungszeichen) gebildet, wobei das erste Zeichen ein Buchstabe sein muss.

In den Befehlen von REDABAS-4 gibt es Positionen, die nur von Feldern belegt werden duerfen, und solche, die nur die Angabe von Speichervariablen gestatten. An vielen Stellen sind aber Ausdruecke erlaubt, die sowohl Felder als auch Speichervariable enthalten duerfen. Alle sichtbaren Speichervariablen und die Felder der im aktiven Arbeitsbereich eroeffneten Datenbankdatei sind ueber ihren blossen Namen ansprechbar. Bei Namensgleichheit gilt i.a. die Vorrangregel

Feld vor Speichervariable,

innerhalb der <ausdrfolge> von DO WITH aber

Speichervariable vor Feld.

Um vor unvorhersehbaren Auswirkungen von Zufaelligkeiten der Namensgebung bei Speichervariablen und Feldern unabhaengig zu sein, sollte einer der folgenden Wege konsequent beschritten werden:

(1) die Namen aller Speichervariablen beginnen mit demselben Zeichen, z.B. S, das als Anfangsbuchstabe aller Felder der Datenbankdateien vereinbarungsgemaess verboten wird

(2) an allen Stellen, wo Ausdruecke zugelassen sind, werden die Speichervariablen ueber ihren Alias M (oder m) qualifiziert:

M-><speichvar> z.B. M->zwert

Die Schreibweise in (2) erinnert an <alias>-><feld> fuer den Zugriff auf Felder aus Datenbankdateien, die in nichtaktivierten Arbeitsbereichen geoeffnet sind. Nur ist dort die Aliasangabe Voraussetzung fuer das Auffinden des Feldes. Spei-

chervariablen sind dagegen bereits ueber ihren Namen im Zugriff. M-><speichvar> wird nur gebraucht, um die oben angefuehrte Vorrangregel fuer zufaellige Namensgleichheit durch eine gezielte Entscheidung zu ersetzen.

Achtung: An solchen Positionen der Befehlssprache, an denen kein Feld zugelassen ist (<speichvar>, <speichvarfolge>), ist die Benutzung von M-> nicht erlaubt.

Beispiel: Erhoehen des Wertes der numerischen Speichervariablen "anz"

| | |
|---|---|
| anz = anz + 1 | ist falsch, wenn ein Feld "anz" im aktivierten Arbeitsbereich anliegt |
| M->anz = M->anz + 1 | ist syntaktisch falsch |
| anz = M->anz + 1 | ist richtig und sicher vor Feld-Einfluessen |

Die Bezugnahme ueber M-> sollte dann gewaehlt werden, wenn Programme entwickelt werden, die nicht auf ganz bestimmte Dateistrukturen zugeschnitten sind, sondern zur allgemeingueltigen Verwendung dienen.

### 8.2.7. Parameteruebergabe und Ergebnisrueckmeldung

Gegenstand der nachfolgenden Betrachtung ist die Uebergabe von Parametern an Programme oder Unterprogramme und die Rueckmeldung der Ergebnisse dieser Programme an die uebergeordnete, aufrufende Ebene.
Beim Entwickeln komplexer oder allgemeingueltig anwendbarer Programme sind technologische Erfordernisse zu beachten:

- weitgehende Entkopplung zwischen Programm und Umgebung: beide duerfen ausser den Anschlussbedingungen keine Detailinformationen voneinander benoetigen;

- gerufene Programme duerfen die Quellen solcher Parameter, die ausschliesslich der Eingabe dienen, nicht verfaelschen.

Der Weg der direkten Entnahme der Parameter aus uebergeordneten (sichtbaren globalen oder lokalen) Speichervariablen und des Eintragens der Ergebnisse in solche hat den Nachteil, dass Programm und Umgebung ueber feste Speichervariablennamen gekoppelt sind, und widerspricht damit den obigen Forderungen.
Deshalb bietet REDABAS-4 eine mit dem Programmaufruf gekoppelte Parameteruebergabe:

```
DO <programmdatei>/<prozedur> WITH <ausdrfolge>
```

Der erste ausfuehrbare Befehl innerhalb des gerufenen Programms muss dann

```
PARAMETERS <parameterfolge>
```

sein.

Beispiel:

```
     .DO progl WITH "ABC", s0, s1+1.27, DATE()
                     |     |     |       |
progl:  PARAMETERS  p1,   p2,   p3,     p4
```

Beim Aufruf durch DO...WITH muessen so viele Ausdruecke folgen, wie das Programm unter PARAMETERS Namen aufgefuehrt hat. PARAMETERS bewirkt implizit ein PRIVATE fuer diese Namen, d.h. sie stehen als lokale Speichervariable bereit. Ihr (Anfangs-) Inhalt entspricht dem Wert der korrespondierenden Ausdruecke beim Aufruf, analog werden Datentyp und typabhaengige Attribute zugewiesen.
Diese Uebergabemethode hat folgende Vorteile:

- keine starre Kopplung von Programm und Umgebung durch feste Namen, das gerufene Programm verwendet ausschliesslich seine unter PARAMETERS vereinbarten Namen und braucht die Quellen der uebergebenen Werte nicht zu kennen;
- Eingabeparameter ("in"-Klasse) koennen neben einzelnen Speichervariablen auch Ausdruecke sein, die waehrend des Aufrufs berechnet werden.

Auch die Rueckmeldung der Ergebnisse kann ueber Parameter ("inout" oder "out"-Klasse) erfolgen. Dazu darf der Parameter aber keine gewoehnliche lokale Variable erzeugen, sondern muss ein Verweis (Referenz) auf eine in der Umgebung existierende (uebergeordnete) Speichervariable bewirken. Ueber den lokal geltenden Namen sind dann Speicherplatz und Attribute der verborgenen aeusseren Variablen im Zugriff.
Die Form eines im DO...WITH-Befehl uebergebenen Ausdrucks entscheidet ueber die Uebergabeart:

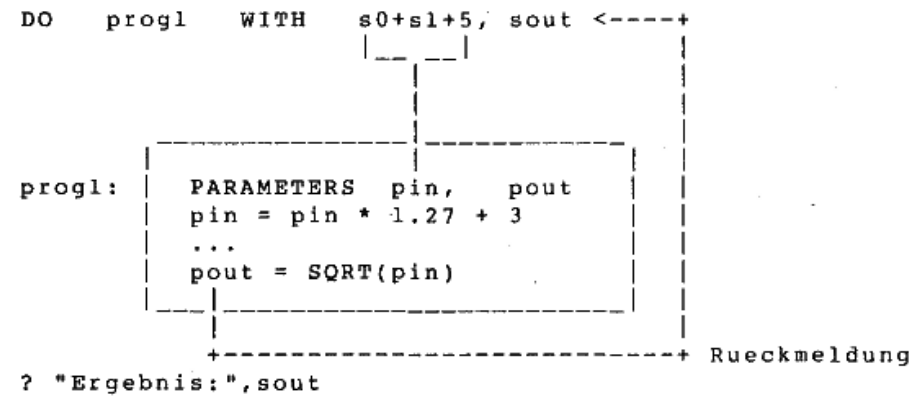
| Art des Uebergabe-Ausdrucks bei WITH | resultierende Uebergabeart und Anwendung |
|---|---|
| <speichvar> | Referenz-Uebergabe ("by reference") fuer "out","inout"(,"in") |
| <feld><br><konstante><br>M-><speichvar><br><alias>-><feld><br>komplexere Ausdruecke | Wert-Uebergabe ("by value") nur fuer "in" |

Wert-Uebergabe hat den Vorteil, dass komplexe Ausdruecke angebbar sind und keine Verfaelschung von Quellvariablen der Umgebung eintreten kann ("in"); Ergebnisrueckmeldungen sind dadurch nicht moeglich.
Referenz-Parameter sind noetig, um eine Rueckmeldung im Parameter zu erlauben. Sie sollten aber nur zu diesem Zwecke verwendet werden ("out", "inout"), weil alle Zuweisungen zu solchen Parametern im gerufenen Programm sich eigentlich in der Quellvariablen ausserhalb des Programms vollziehen.

Je nachdem, ob der zur Rueckmeldung bestimmte Parameter auch eine Eingabebedeutung hat ("inout") oder nicht ("out"), ergeben sich zwei Arbeitsprinzipien, die im folgenden Beispiel skiz-

ziert sind. s0, s1, sout und sinout sollen dabei numerische Speichervariable bezeichnen.

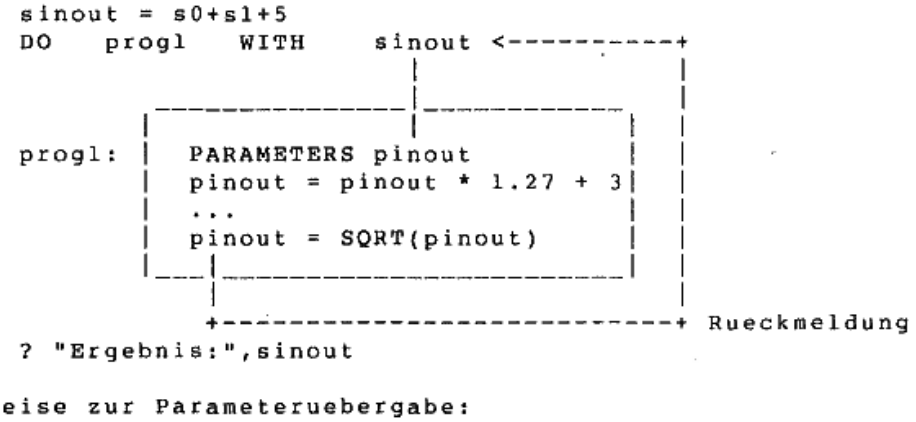
(1) getrennte Rueckmeldung




```
? "Ergebnis:",sout
```

(2) Rueckmeldung ueberschreibt die Eingabe

```
sinout = s0+s1+5
```




```
? "Ergebnis:",sinout
```

Hinweise zur Parameteruebergabe:
- Zur Wertuebergabe einer einzelnen Speichervariablen ist die Angabe von M-><speichvar> statt <speichvar> bei WITH erforderlich.
- Felder sollten immer mit Alias als <alias>-><feld> spezifiziert werden, da in <ausdrfolge> bei WITH der Vorrang umgekehrt ist.
- Referenz-Uebergabe sollte nur am Ende der Parameterliste vorgesehen werden, da die als Quelle dienende Speichervariable sofort verborgen wird. Sie ist auch im Rest der <ausdrfolge> bei WITH nicht mehr ueber ihren Namen ansprechbar, z.B. ist DO prog WITH s1,s2-s1 im Unterschied zu DO prog WITH s2-s1,s1 nicht ausfuehrbar.

### 8.2.8. Allgemein verwendbare Programme

Die vielen zu Speichervariablen dargestellten Details, die die Attribute und die Parameteruebergabe betreffen, sind vor allem fuer versierte Anwendungsprogrammierer von Bedeutung. Auf das Erstellen einfacher, an ein festes Umfeld gebundener Programme ergeben sich kaum Auswirkungen.
Sollen aber allgemeingueltig einsetzbare Programme entworfen werden, sind die folgenden Richtlinien zu beachten:

1. Schutz vor der implizit wirkenden Vorrangregel bei gleichnamigen Feldern und Speichervariablen:
   Speichervariable in Ausdruecken durch M-> praezisieren,
   Felder bei DO WITH durch <alias>-> praezisieren

2. Schutz vor Namenskonflikten der Speichervariablen untereinander:
   alle lokal benoetigten Speichervariablen vor ihrer ersten Zuweisung im Programm mit PRIVATE deklarieren

3. Sichern der Quellen von uebergebenen "in"-Parametern vor Verfaelschung durch das gerufene Programm:
   - entweder im gerufenen Programm keine Zuweisungen auf Parameter ausfuehren, die nicht zur Rueckmeldung dienen (Arbeiten mit anderen lokalen Variablen nach einem Kopieren der Parameterinhalte in diese)
   - oder im rufenden Programm die Wert-Uebergabe durch M-> erzwingen

Zu jedem [Unter-] Programm sollten die Anschlussbedingungen sorgfaeltig dokumentiert werden, u.a.:
- Anzahl und Bedeutung der Parameter,
- geforderter Datentyp bzw. zulaessige Typvarianten (falls eine Analyse mittels TYPE("...") vorgesehen ist),
- Forderungen an die Uebergabe-Art, falls die Parameter zur Rueckmeldung von Ergebnissen bestimmt sind (Referenz-Uebergabe),
- Datentyp bzw. Typvarianten der Rueckmeldung.

Beispiel eines allgemeingueltigen Programms "maxC":

```
PARAMETERS feld
* feld:      Name eines Feldes vom Datentyp Zeichen der akti-
*            vierten Datenbankdatei
* Aufruf:    DO maxC WITH "<feldC>"
* Funktion:  Durchsuchen der Datei nach dem Satz, in dem das
*            Feld <feldC> vom Datentyp Zeichen die laengste
*            Belegung besitzt
* Ergebnis:  Anzeige der Nummer des Satzes, der belegten Laen-
*            ge des Feldes vom Datentyp Zeichen und des Satzes
*            selbst.
SET ECHO OFF
SET TALK OFF
*
GO TOP
PRIVATE maxr,maxl
*
IF TYPE("M->feld") # "C"  && Parameter-Typ pruefen
  ? "*** Unerlaubter Datentyp des Parameters"
  SET TALK ON
  RETURN
ENDIF
*    ...       hier spaeter Einschub ------
IF TYPE("&feld") # "C"    && Datentyp des Feldes pruefen
  ? "*** Kein Feld oder Datentyp ungleich C"
  SET TALK ON
  RETURN
ENDIF
*
```

```
STORE  0  TO maxr, maxl
DO WHILE .NOT. EOF()       && Datei durchsuchen
  IF LEN(TRIM(&feld)) > M->maxl
    maxl = LEN(TRIM(&feld))
    maxr = RECNO()
  ENDIF
  SKIP
ENDDO
? "recno=",M->maxr,"maxl=",M->maxl      && Ergebnis
DISPLAY RECORD M->maxr
*
SET TALK ON
RETURN
```

Das angegebene Programm arbeitet nicht korrekt, wenn der im WITH - Parameter uebergebene Name nicht mit einem Feldnamen der aktuellen Datenbankdatei, dafuer aber zufaellig mit dem Namen einer sichtbaren Speichervariablen vom Typ C uebereinstimmt. Die mit der TYPE()-Funktion ausgefuehrte Pruefung auf den Datentyp des Feldes ist noch unvollkommen. Zur Klaerung folgende Tabelle:

| Variable <var> vorhanden als Feld | Variable <var> vorhanden als Speichvar. | Pruef-Funktion | Ergebnis |
|---|---|---|---|
| X | X | TYPE("<var>") | Datentyp Feld |
| X | - | | Datentyp Feld |
| - | X | | Datentyp Speichvar. |
| - | - | | "U" |
| X | X | TYPE("M-><var>") | Datentyp Speichvar. |
| X | - | | "U" |
| - | X | | Datentyp Speichvar. |
| - | - | | "U" |
| X | X | TYPE("<alias>-><var>") | Datentyp Feld |
| X | - | | Datentyp Feld |
| - | X | | "U" |
| - | - | | "U" |

Bei der Pruefung auf den Datentyp des Feldes muss also der Alias der Datenbankdatei/des Arbeitsbereichs angegeben werden, um sicher zu sein, dass keine zufaellig definierte Speichervariable stoert:

```
IF TYPE("<alias>->&feld") # "C"
```

Soll aber das Programm die Kenntnis des Alias nicht voraussetzen, dann muss "&feld" zunaechst auf Uebereinstimmung mit einem der Feldnamen geprueft werden, wozu die FIELD()-Funktion dient. Im obigen Programm ist an der gekennzeichneten Stelle folgender Einschub noetig:

```
PRIVATE sn                    && Feld-Existenz pruefen
sn=1
DO WHILE "" # FIELD(M->sn)
  IF UPPER(feld) = FIELD(M->sn)
    EXIT                      && Feld gefunden
  ENDIF
  sn=sn+1
ENDDO
IF "" = FIELD(M->sn)
  ?"*** Kein Feld"
  SET TALK ON
  RETURN
ENDIF
```

wobei die beim nachfolgenden Datentyp-Test ausgegebene Nachricht jetzt

```
  ?"*** Datentyp des Feldes ungleich C"
```

lauten kann.
Weiterhin koennte das Programm am Anfang mit der DBF()-Funktion testen, ob ueberhaupt eine Datenbankdatei geoeffnet ist.
Wie das angefuehrte Beispiel zeigt, laesst sich die besondere Programmqualitaet, in einem unbekannten Umfeld stoerungsfrei zu arbeiten, nur mit viel Sorgfalt und Ueberlegung erreichen.

## 8.3. Befehlssyntax und Ausdruecke

### 8.3.1. Syntax

#### 8.3.1.1. Syntax der Befehle

Ein Befehl beginnt mit dem Befehlswort. Mitunter ist dieses Befehlswort schon ausreichend, um REDABAS-4 zu einer sinnvollen Aktion zu veranlassen. Gewoehnlich wird sich ein Befehl jedoch aus dem Befehlswort und einer oder mehrerer Zusatzklauseln zusammensetzen.
Eine Zusatzklausel besteht aus einem Einleitungswort und haeufig einer variablen Angabe (Dateispezifikation, Feldname, Bedingung, Ausdrucksfolge usw.).

Die Befehle verwenden englische Steuerwoerter und sind in ihrer Struktur weitgehend Saetzen der natuerlichen Sprache angepasst. Deshalb ist die Syntax aller REDABAS-4 - Befehle auch nicht durch ein einziges Schema darstellbar. Nahezu alle Befehle entsprechen aber in ihrer Syntax einer der folgenden groben Darstellungen:

```
<befehlswort> <datei> [Zusatzklauseln]
<befehlswort> <ausdrfolge> [Zusatzklauseln]
<befehlswort> [Zusatzklauseln] <ausdrfolge>
```

Das <befehlswort> ist dabei nicht immer ein einziges Wort. Mitunter ist eine Wortgruppe noetig, um einen Befehl eindeutig zu identifizieren, z.B. MODIFY COMMAND oder SET RELATION TO.

Symboldefinitionen, wie z.B. <ausdrfolge> sind im Abschnitt 9.1. erklaert.
Die genaue Beschreibung aller Befehle erfolgt im Abschnitt 9.3.3.

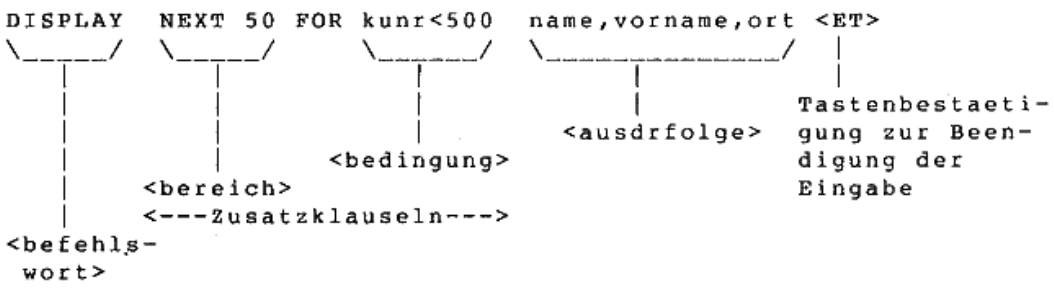
Allgemeine Hinweise fuer die Notation von REDABAS-4-Befehlen und die Grundregeln der REDABAS-4-Sprache sind in den Abschnitten 2.2. und 9.3.1. zu finden.

Viele REDABAS-4 - Befehle sind mit einer Datensatzauswahl verbunden und benutzen dazu die Zusatzklauseln

[<bereich>] [WHILE <bedingung>] [FOR <bedingung>]

(siehe Abschnitt 9.3.2.).

Ein REDABAS-4-Befehl kann beispielsweise folgendermassen aussehen:




### 8.3.1.2. Syntax der Ausdruecke

Die variablen Angaben innerhalb oder ausserhalb der Zusatzklauseln eines REDABAS-4 - Befehls sind haeufig Ausdrucksfolgen. Eine Ausdrucksfolge <ausdrfolge> besteht aus einem einzelnen Ausdruck <ausdr> oder mehreren durch Komma voneinander getrennten Ausdruecken:

<ausdrfolge>: <ausdr>[,<ausdr>[,...]]

Ein Ausdruck kann ein einzelnes Datenelement, aber auch eine Verknuepfung (siehe Abschnitt 8.3.3.) und/oder Funktion (siehe Abschnitt 9.2.) von Datenelementen sein. Die Auswertung eines Ausdrucks ergibt einen einfachen Wert, der vom Datentyp her numerisch (N), logisch (L), ein Datum (D) oder eine Zeichenreihe (C) sein kann.

Datenelemente werden in der REDABAS-4 - Befehlssprache dargestellt durch:

- Variablennamen (repraesentieren den momentanen Variablen-Inhalt, der ausserhalb des Befehls in einer Speicherzelle enthalten ist)
  - Feldnamen (repraesentieren den momentanen Feldinhalt im aktuellen Datensatz einer geoeffneten Datenbankdatei)
  - Speichervariablennamen (repreaesentieren den momentanen Inhalt der Speichervariablen)

- Konstanten (fuegen fest vorgegebene Werte direkt in den Befehl ein)

Im folgenden wird sprachlich kaum noch unterschieden zwischen dem Namen einer Variablen (der im Ausdruck innerhalb des Befehls angegeben wird und die Variable identifiziert) und dem momentanen Inhalt der Variablen, den dieser Name repraesentiert. Stattdessen wird einfach von Feldern und Speichervariablen gesprochen.
Unter Inhalt einer Variablen soll nicht allein der Wert verstanden werden. Dazu gehoeren auch alle weiteren fuer die Verknuepfung und Auswertung wesentlichen Attribute, vor allem der Datentyp.

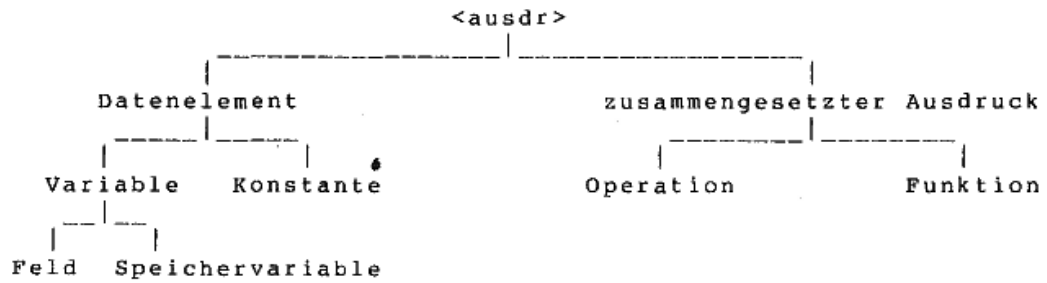
Fuer die Verknuepfung von Datenelementen, fuer die Umwandlung und Berechnung von Werten, bietet REDABAS-4

- Operationen (siehe Abschnitt 8.3.3.)
- Funktionen (siehe Abschnitt 9.2.).

Komplexe Ausdruecke sind moeglich, d.h. die Anwendung von Operationen oder Funktionen auf Ausdrucke liefert wieder einen Ausdruck.

In dieser Programmtechnischen Beschreibung wird ein beliebiger Ausdruck als <ausdr> dargestellt. Soll ein bestimmter Datentyp spezifiziert werden, geschieht das durch Anfuegen eines der Buchstaben C, D, L oder N, wie z.B. <ausdrN> fuer einen numerischen Ausdruck.
Logische Ausdruecke (<ausdrL>) werden auch als Bedingung (<bedingung>) bezeichnet und haben grosse Bedeutung bei der Datensatz-Auswahl (Recherche in Datenbankdateien) und bei der Steuerung von Programmschleifen.

Die Syntax eines Ausdrucks laesst sich zusammengefasst darstellen als:




Weiterhin gilt:

| | | |
|---|---|---|
| Operation: | <ausdr1><operator><ausdr2> | oder |
| | <operator><ausdr> | |
| Funktion: | <funktion>([<ausdrfolge>]) | |

Nur fuer diese Erklaerung, und darum nicht im Abschnitt 9.1. erwaehnt, wurden dabei folgende Symbole eingefuehrt:

| | |
|---|---|
| <operator> | Bezeichnung eines Operators, der eine Operation ausloest, z.B. + |
| <funktion> | Name einer Funktion, z.B. STR (siehe Abschnitt 9.2.) |

## 8.3.2. Variablen und Konstanten

### 8.3.2.1. Variablen

Variablen werden in REDABAS-4 stets mit einem Namen bezeichnet. Ihr Inhalt (der Wert, mitunter auch weitere Datenattribute) ist veraenderlich. Steht der Variablenname in einem Ausdruck der REDABAS-4 - Befehlssprache, dann dient der momentane Inhalt der Variablen zur Auswertung des Ausdrucks.

REDABAS-4 unterscheidet Datenbankvariablen und Speichervariablen. Datenbankvariablen sind alle durch den Feldnamen bezeichneten Felder eines aktuellen Datenbanksatzes. Speichervariablen dienen als Zwischenspeicher und sind weder einer Datenbankdatei noch einem Datensatz zugeordnet.

Felder:

In Feldern koennen Daten aller 5 in REDABAS-4 verfuegbaren Datentypen gespeichert werden (s. Abschn. 8.1.1.).

Zur Benutzung und Verknuepfung in Ausdruecken eigenen sich nur
- Zeichenfelder (<feldC>, Datentyp C)
- numerische Felder (<feldN>, Datentyp N)
- Datumsfelder (<feldD>, Datentyp D)
- logische Felder (<feldL>, Datentyp L)

Der Feldinhalt wird bei Datentypen wie N und D bei der Auswertung von Ausdruecken zunaechst in eine geeignetere interne Darstellung konvertiert.

Merkfelder (Datentyp M) duerfen in REDABAS-4 nicht als Ausdruck oder in Ausdruecken verwendet werden (siehe auch Abschnitt 3.6.).

Speichervariablen:

Speichervariablen (siehe Abschnitt 8.2.) sind in den Typen
- Zeichen (<speichvarC>, Datentyp C)
- numerisch (<speichvarN>, Datentyp N)
- Datum (<speichvarD>, Datentyp D)
- logisch (<speichvarL>, Datentyp L)

moeglich.

Speichervariablen dienen der Zwischenspeicherung von Daten ausserhalb der Struktur von Datenbankdateien. Eine Speichervariable ist durch ihren Namen, Datentyp, Wert, Existenzbereich und die Sichtbarkeit gekennzeichnet. Alle diese Charakteristika werden im Arbeitsspeicher gefuehrt.

### 8.3.2.2. Konstanten

Konstanten sind im Sinne von REDABAS-4 unveraenderliche Direktwerte, die nicht durch einen Namen bezeichnet, sondern selbsterklaerend in einen Befehl eingefuegt werden.

Bezueglich des Typs unterscheidet man bei Konstanten Zeichen (Datentyp C), Numerisch (Datentyp N) und Logisch (Datentyp L).

Zeichenreihenkonstanten muessen stets in Hochkommas ' ', Anfuehrungszeichen " " oder eckige Klammern [] eingeschlossen sein. Enthaelt eine Zeichenreihe eines dieser Zeichen, so ist die Konstante durch ein anderes der verbleibenden Sonderzeichen einzuschliessen. In dieser Programmtechnischen Beschreibung werden bevorzugt Anfuehrungszeichen verwendet, waehrend eckige Klammern leicht zu Verwechslungen mit der Kennzeichnung auslassbarer Befehlsbestandteile fuehren koennten.

Beispiele fuer Zeichenreihenkonstanten sind: [AUTO], 'Auto[s]', "a'b'c'd".

Numerische Konstanten sind Zahlen, die Vorzeichen und Dezimalpunkt haben koennen (z. B. 1, -25, +21.7).

Logische Konstanten bezeichnen Wahrheitswerte und werden durch REDABAS-4 wie folgt dargestellt:
.T.,.t., .Y.,.y. steht fuer "wahr"
.F.,.f., .N.,.n. steht fuer "falsch".
(Die einschliessenden Punkte sind fuer das Identifizieren des Konstantentyps noetig.)

```
Beispiel:    : STORE .Y. TO Slog
---------    .T.
```

Bei der Dateneingabe in eine Maske (z.B. APPEND) sind die logischen Werte nicht als Konstanten, sondern als Buchstaben ohne einschliessende Punkte anzugeben. Anstelle der Buchstaben Y bzw. y steht dafuer J bzw. j.

Konstanten vom Typ "Datum" sind nicht vorgesehen.
Mit DATE() oder CTOD() (beide Funktionen siehe Abschnitt 9.2.) kann jedoch ein konkretes Datum als Datentyp "D" bereitgestellt werden.

```
Beispiel:    : STORE CTOD ("16.11.89") TO Sdat
---------    16.11.89
             : ? TYPE("Sdat")
             D
```

### 8.3.3. Operationen

REDABAS-4 ermoeglicht folgende Operationen:

- arithmetische Operationen (Berechnungen)
- Vergleichsoperationen
- logische Operationen
- Zeichenreihen-Operationen
- Datumsoperationen.

In einer Operation sind die Operanden durch Operatoren verknuepft. Ergebnisse von Operationen koennen selbst wieder Operanden sein. Zu diesem Zweck sind die Operationen ggf. in Klammern einzuschliessen. Durch Klammerung wird eine gewuenschte Ausfuehrungsreihenfolge von Operationen erreicht.

Alle Operationen unterliegen einer strengen Datentyppruefung, d.h. fuer die Operanden sind nur bestimmte Datentypkombinationen zulaessig. Die nachfolgende Tabelle zeigt die Datentypan-

forderungen aller Operationen.

| Operation | Datentypanforderung | Erklaerung |
|---|---|---|
| arithmetisch | N X N -> N | X: +,-,*,/,** oder ^ |
| Vergleich | N X N -> L<br>D X D -> L<br>C X C -> L<br>C $ C -> L | X: =,# oder <>,<,>,<=,>=<br><br>"enthalten in" |
| Datum | D - N -> D<br>D + N -> D<br>N + D -> D<br>D - D -> N | Datumsarithmetik |
| Zeichenreihe | C + C -> C<br>C - C -> C | Zeichenreihen-kettung |
| logisch | .NOT. L -> L<br>L .OR. L -> L<br>L .AND. L -> L | |

Bei der Formulierung von Operationen ist auf die Typengleichheit der Operanden (bzw. ihre Kombinierbarkeit) und die Vertraeglichkeit von Operandentyp und Operator zu achten. So kann z.B. nur eine Zeichenreihe mit einer anderen Zeichenreihe verknuepft werden, nicht aber eine Zeichenreihe mit einem numerischen Wert. In diesem Zusammenhang sei auf die Moeglichkeiten der Typumwandlung mittels Funktionen verwiesen (siehe Abschnitt 9.2.).

Operanden und Operatoren koennen lueckenlos aneinandergefuegt oder durch Leerzeichen voneinander getrennt sein.
Beispiele fuer Operationen:

```
: ? 3+8
11

: ? "3"+"a"
3a

: ? 3+"a"
      verursacht eine Fehlernachricht wegen Datentyp-
      Unvertraeglichkeit der Operanden
```

### 8.3.3.1. Arithmetische Operationen (Berechnungen)

Die Operanden von Berechnungen sind -ebenso wie die Ergebnisse- von numerischem Datentyp. Als arithmetische Operationen sind die vier Grundrechenarten sowie Potenzierung zugelassen und durch folgende Operatoren anzugeben:

| | |
|---|---|
| + | Addition |
| - | Subtraktion |
| * | Multiplikation |
| / | Division |
| ** oder ^ | Potenzierung |

() Die Klammern dienen zur Festlegung der Bearbeitungsreihenfolge.

Bei der Ausfuehrung der Operationen werden zunaechst Klammerausdruecke aufgeloest. Geschachtelte Klammern werden in der Reihenfolge von innen nach aussen berechnet.
Fuer die Rangfolge innerhalb der Operatoren gilt
1. Vorzeichen (+,-)
2. Potenzierung
3. Multiplikation und Division
4. Addition und Subtraktion.

Bei gleichrangigen Operatoren erfolgt die Auswertung von links nach rechts.

Der Wertebereich fuer Berechnungen und fuer die Darstellung der Ergebnisse in Speichervariablen bewegt sich in der Groessenordnung

```
        +99        -307          -307        +99
    -10   ... -10     , 0 , +10     ... +10
```

Die Genauigkeit betraegt 13 bis 15 Ziffern, wobei in der Naehe der Wertebereichsgrenzen Genauigkeitsverluste auftreten. Bei numerischem Ueberlauf wird das Ergebnis durch eine Reihe von Sternen angezeigt, in manchen Faellen auch als Gleitkommazahl (Achtung: Es ist noch keine durchgaengige Gleitkomma-Unterstuetzung installiert.).
Fuer Berechnungen wird die hoechstmoegliche intern gespeicherte Genauigkeit genutzt, auch wenn eine externe Anzeige auf wenige Stellen gerundet erscheint.

Beispiele:

```
: ? 3+4*5 , (3+4)*5
  23    35

: ? ((9+4)*3+1)*2, (9+4)*3+1*2, 9+4*3+1*2
     80     41     23

: ? 2^3
          8.00

: ? 5/3
 1.67

: ? 10 + 7.345
 17.345
```

```
: ? 2.31 * 4.567
 10.54977
```

### 8.3.3.2. Vergleichsoperationen

Numerische, Zeichenreihen- und Datumsausdruecke lassen sich vergleichen, wobei jeweils nur der gleiche Typ innerhalb einer Vergleichsoperation auftreten darf.

Das Ergebnis einer Vergleichsoperation ist stets ein logischer Wert, .T. fuer "wahr" und .F. fuer "falsch".

Folgende Operatoren sind moeglich:

| | |
|---|---|
| < | kleiner als |
| > | groesser als |
| = | gleich |
| <> oder # | ungleich |
| <= | kleiner oder gleich |
| >= | groesser oder gleich |
| $ | enthalten in (Teilzeichenreihen-Vergleichsoperator) |

Der Teilzeichenreihen-Vergleichsoperator "$" ist nur fuer Zeichenreihen verwendbar und testet, ob die erste Zeichenreihe in der zweiten enthalten ist oder gar mit ihr uebereinstimmt. Dieser Vergleich wird auch als gleitendes Suchen bezeichnet. Das Ergebnis ist wie bei allen Vergleichen ".T." oder ".F.". Im Unterschied dazu liefert die AT()-Funktion (s. Abschn. 9.2.) die Position innerhalb der zweiten Zeichenreihe als numerischen Wert.

Hinweise zum Zeichenreihenvergleich:
Der Vergleich von Zeichenreihen erfolgt von links nach rechts, wobei in der Laenge des zweiten Operanden (standardmaessig, vgl. SET EXACT im Abschn. 9.3.3.) verglichen wird. Die Rangfolge der Zeichen weicht etwas von der Zeichenverschluesselung im ASCII-Code ab, die aufsteigende Reihenfolge (Zeichen-Sortierfolge) lautet: Leerzeichen, Ziffern, Grossbuchstaben, Kleinbuchstaben, Sonderzeichen (in sich gruppiert: Klammerungszeichen, Vergleichssymbole, Zeichen fuer mathematische Operationen, Interpunktionszeichen usw.). Die Umlaute werden in Nachbarschaft der entsprechenden Vokale eingeordnet, "sz" bei s.
Ist die beim Vergleich inbegriffene Unterscheidung zwischen Gross- und Kleinbuchstaben nicht erwuenscht, muessen die Vergleichsoperanden der UPPER-() bzw. LOWER-()Funktion unterworfen werden.

Wenn der EXACT-Schalter auf OFF steht, hat die Reihenfolge Bedeutung, in der die beiden Operanden bei der Vergleichsformulierung aufgefuehrt werden. Das weicht von der mathematischen Interpretation eines Vergleichs ab, bringt aber Vorteile in der Datenverarbeitungspraxis, z.B. bei der Suche ueber Schluessel. Eine leere Zeichenreihe (Laenge 0, z.B. "" oder CHR(0)) darf nicht rechts vom Vergleichsoperator stehen, weil dann immer Gleichheit angezeigt wird. Statt der Bedingung svar = "" ist entweder "" = svar oder LEN(svar) = 0 zu schreiben.

Die folgende Tabelle zeigt die Wirkung des SET EXACT - Schalters an einem systematisch dargestellten Beispiel:

| 1.Operand | 2. Operand "AB " | "ABCD" | "AB" |
|---|---|---|---|
| "AB " | .T. | .F. | .T. |
| "ABCD" | .F. | .T. | .T./.F. |
| "AB" | .F./.T. | .F. | .T. |

fuer SET EXACT OFF / ON

| | "" | " " | "AB" |
|---|---|---|---|
| "" | .T. | .F./.T. | .F. |
| " " | .T. | .T. | .F. |
| "AB" | .T./.F. | .F. | .T. |

(Fuer SET EXACT ON ist die Matrix symmetrisch, weil die Operandenreihenfolge dann keine Auswirkungen hat.)

Hinweise zum numerischen Vergleich:
Beim Vergleich numerischer Daten ist zu beachten, dass die Genauigkeit der Anzeige einer Zahl geringer sein kann als die intern gefuehrte Genauigkeit, mit der ein Berechnungsergebnis vorliegt oder in einer Speichervariablen dargestellt ist.

Da Zahlen intern in binaerer Gleitkommadarstellung benutzt werden, sind bei Dezimalzahlen mit gebrochenem Anteil Darstellungsungenauigkeiten moeglich, die sich bei fortgesetzter Anwendung von Operationen und Funktionen aufschaukeln koennen. Damit dieser Effekt den numerischen Vergleich nicht stoert, werden zum Vergleichen (ausser mit 0) hoechstens 13 Stellen herangezogen.
Nach groessenordnungsmaessig 1000 Rechenoperationen (ohne zwischenzeitliche Konvertierung ins Dezimale) kann sich im unguenstigsten Falle die Darstellungsungenauigkeit trotzdem auf den Vergleich auswirken.
Deshalb muesste die interne Zahl nach etwa 1000 Rechenoperationen sicherheitshalber wieder dezimal normiert werden. Sie kann entweder durch REPLACE ins Datenbankfeld eingetragen und wieder in die Speichervariable geladen werden oder durch Anwendung von Funktionen normiert werden. Zu Letzterem ein Beispiel: Mit

```
sgpreis = VAL(STR(sgpreis,12,2))
```

oder

```
sgpreis = ROUND(sgpreis,2)
```

wird eine Speichervariable "sgpreis" normiert, die einen Gesamtpreis (2 Dezimalstellen) enthalten soll.

Hinweis zum Datumsvergleich:
Beim Datumsvergleich wird das laufende Datum als von Tag zu Tag gleichfoermig wachsend betrachtet, so dass eine korrekte zeitliche Ordnung gewaehrleistet ist.

Hinweise zum logischen Datentyp:
Vergleichsoperationen liefern Ergebnisse vom Datentyp logisch. Andererseits duerfen auf den logischen Datentyp keine Vergleichsoperationen angewandt werden. Das ist bei der Bildung von logischen Ausdruecken (Bedingungen) zu beachten. Datenfelder oder Speichervariablen, die diesen Datentyp besitzen, stellen schon einen kompletten logischen Ausdruck dar und koennen nur durch logische Operatoren (.AND., .OR., .NOT.) verknuepft werden. Die folgende Tabelle illustriert das fuer zwei

logische Felder "L1" und "L2" einer Datenbankdatei:

```
Bedingung :
 falsch    | richtig                                    Beispiel
-----------|-------------------------------------------------------
 L1 = .T.  | L1                                         DO WHILE L1
 L1 = .F.  | .NOT. L1                                   LIST FOR .NOT.L1
 L1 = L2   | L1.AND.L2 .OR. .NOT.L1.AND..NOT.L2
```

Beispiele:

```
 : ? "MOND" < "SONNE"
 .T.
 : ? 3> -5
 .T.
 : ? "ABC" > "abc"
 .F.
 : ? "ab" = "abc"
 .F.
 : ? "abc" = "ab"
 .T.    (falls EXACT-Schalter OFF, vgl. SET EXACT in
         Abschn. 9.3.3.)
 : ? "ERNST" $ "BERNSTEIN"
 .T.
```

### 8.3.3.3. Logische Operationen

Solche Operationen sind nur auf Operanden vom Datentyp logisch anwendbar. Das Ergebnis ist ebenfalls ein logischer Wert.

Die moeglichen Operatoren in der Reihenfolge ihres Vorrangs sind:

| | |
|---|---|
| .NOT. | logische Verneinung |
| .AND. | logisches UND |
| .OR. | logisches ODER |

Die Verwendung von Klammern ist moeglich, sie stehen in der Rangfolge der Abarbeitung ueber den anderen Operatoren.
Der .NOT.-Operator ist einstellig, d.h. er bezieht sich nur auf einen Operanden. Die logische Verneinung kehrt den Wahrheitswert des Operanden um.
Das logische UND ergibt den Wahrheitswert .T. nur dann, wenn beide Operanden "wahr" sind.
Das logische ODER bringt .T. als Ergebnis, wenn mindestens einer der Operanden "wahr" ist.

Beispiele:

```
 : STORE .f. TO sv1
 .F.
 : sv2 = .y.
 .T.
 : ? sv1 .OR. sv2 , sv1.AND.sv2
 .T. .F.
 : ? .NOT.sv1 .AND. sv2
 .T.
```

### 8.3.3.4. Zeichenreihen-Operationen

Zeichenreihen-Operationen werden auf Zeichenreihen angewendet. Das Ergebnis entsteht durch Aneinanderfuegen der Zeichenreihen und ist wieder eine Zeichenreihe. Die beiden moeglichen Operatoren unterscheiden sich wie folgt:

+ Zeichenreihen werden in der Reihenfolge ihrer Nennung in voller, definierter Laenge aneinandergefuegt.

- Zeichenreihen werden in der Reihenfolge ihrer Nennung aneinandergefuegt, wobei Leerzeichen vom Ende der links vom Operator stehenden Zeichenreihe an das Ende der Ergebniszeichenreihe verlagert werden.

Der "minus"-Operator "-" tilgt keine Leerzeichen, sondern lagert nachgestellte Leerzeichen an das Ende der entstandenen Zeichenreihe um. Fuehrende oder eingebettete Leerzeichen innerhalb der Zeichenreihe bleiben unveraendert. Fuer beide Operatoren gilt: Die Laenge des Ergebnisses ist die Summe der Laenge der verknuepften Zeichenreihen.

Die Beseitigung der Leerstellen vom Ende einer Zeichenreihe kann mit der Funktion TRIM() oder RTRIM() erfolgen (siehe Abschnitt 9.2.) waehrend LTRIM() vorangestellte Leerzeichen entfernt.

Wenn mehr als zwei Zeichenreihen verkettet werden und der "-"-Operator beteiligt ist, wird das Ergebnis nur durch schrittweise Betrachtung von links nach rechts verstaendlich. Ausserdem sind Klammerungen zur Veraenderung dieser Auswertungsreihenfolge gestattet.

Beispiele:

```
: STORE "Max     " TO svname
Max
: STORE "Meier   " TO sname
Meier
: ? svname + sname
Max     Meier
: ? svname - sname
MaxMeier
: ? TRIM(svname)+" "+sname
Max Meier
: ? sname - "," - svname
Meier,Max
: ? svname-(" "+sname)
Max Meier
: ? sname-","-(" "+svname)
Meier, Max
```

### 8.3.3.5. Datumsoperationen

REDABAS-4 bietet fuer Ausdruecke vom Datentyp "Datum" zwei Moeglichkeiten der Berechnung:

- Berechnen eines neuen Datumswerts aus einem anderen Datumswert durch Addieren oder Subtrahieren einer Anzahl von Tagen

- Bildung der Differenz (in Tagen) von zwei Datumsangaben. Das Ergebnis ist numerisch.

Es sind die Operatoren

```
+  Addition
-  Subtraktion
```

moeglich.

Beispiele:

```
: ? DATE()
31.12.88
: ? DATE() + 100
10.04.89
: ? DATE() - 365
01.01.88
: ? DATE() - CTOD("24.06.81")
      2747
```

### 8.3.3.6. Rangfolge von Operationen

Sind in einem Ausdruck mehrere Operationen enthalten, so erfolgt die Auswertung in einer definierten Reihenfolge.

Hierbei gelten Vorrangregeln, die entscheiden, welche Operation vor einer anderen ausgefuehrt wird. Treten gleichrangige Operationen nebeneinander auf, so wird die Abarbeitung von links beginnend ausgefuehrt.

In der nachfolgenden Uebersicht sind die Operationen bzw. Operatoren ihrer Wertigkeit entsprechend aufgelistet, beginnend mit den ranghoechsten. Gleichrangige Operationen sind nebeneinander angeordnet.

- Klammern
- Funktionen
- Vorzeichen (+, -)
- Potenzierung
- Multiplikation, Division (*, /)
- Addition, Subtraktion (+, -) fuer Datentypen N und D
- Zeichenreihenoperationen (+, -)
- Vergleiche (<, >, =, <>, <=, >=, $)
- logische Verneinung (.NOT.)
- logisches UND (.AND.)
- logisches ODER (.OR.)

Beispiele:
---

Die Beispiele sind so gruppiert, dass zunaechst keine Klammern im Ausdruck benutzt werden. Anschliessend sind ohne sonstige Aenderung Klammern eingesetzt worden, zur Demonstration aber genau so, wie die Auswertung auch unter reiner Steuerung der Rangfolge ablaeuft. Mitunter ist noch gezeigt, dass andere Klammerungen das Ergebnis i.a. modifizieren.

```
: ? 1+2*3^4 = 1+(2*(3^4)) , 1+2*3^4
.T.             163.00

: ? "a"+"b " $ "xa  "-"b"
.T.
: ?("a"+"b ")$("xa  "-"b")
.T.

: ? 3 = 4-1  .AND..T.
.T.
: ?(3 =(4-1)).AND..T.
.T.

: ? .NOT. .T. .AND. .F.
.F.
: ?(.NOT. .T.).AND. .F.
.F.
: ? .NOT.(.T. .AND. .F.)
.T.

: ? .F. .AND. .F. .OR. .T. .AND. .T.
.T.
: ?(.F. .AND. .F.).OR.(.T. .AND. .T.)
.T.
: ? .F. .AND.(.F. .OR. .T.).AND. .T.
.F.
: ? .F. .AND.(.F. .OR.(.T. .AND. .T.))
.F.
```

## 8.4. Hinweise zur Dialogfuehrung

### 8.4.1. Fehlerbehandlung

#### 8.4.1.1. Interaktiver Modus

Mit dem Bereitschaftszeichen ": " auf dem Bildschirm fordert REDABAS-4 den Nutzer zur Eingabe eines Befehls auf. Durch Betaetigung der <ET>-Taste (bzw. <ENTER>, <ETl>, <RETURN>, <--') wird die Befehlseingabe abgeschlossen. Bemerkt der Nutzer waehrend der Eingabe Tippfehler, kann er diese entweder durch die Korrekturtaste (Rueckwaertspfeil) und die vom REDABAS-4-Texteditor fuer das Editieren von Zeilen bekannten Tastenkombinationen verbessern oder mit der <Esc>-Taste die gesamte Zeile loeschen und neu eingeben.
Nach erfolgter Eingabe prueft REDABAS-4 die Befehlszeile auf korrekte Syntax und Semantik. Stellt das erste Wort kein gueltiges Befehlswort dar (es genuegt die Eingabe der ersten vier Zeichen des Befehlswortes), so erscheint die Bildschirmnachricht

```
*** Unbekannter Befehl.
```

Auf der naechsten Zeile folgt ein Fragezeichen. Darunter wird der fuer REDABAS-4 "unbekannte Befehl" nochmals ausgeschrieben und REDABAS-4 fragt

```
Wuenschen Sie Hilfe? (J/N)
```

Entscheidet man sich fuer "J", wird in das "REDABAS-4-Hauptmenue" des HELP-Befehls verzweigt, und es koennen wunschgemaess Informationen eingeholt und anschliessend das korrekte Befehlswort eingegeben werden.
Entscheidet sich der Nutzer nicht fuer eine Hilfestellung, wird N oder <ET> gedrueckt und die Eingabe wiederholt.
Nach Pruefung des Befehlswortes erfolgt die Syntaxkontrolle der Befehlszeile. Erkennt REDABAS-4 einen Fehler, erscheint die Meldung

```
Syntaxfehler.
```

und ausserdem wird ueber das als ungueltig angesehene Wort (hinter dem Wortende positioniert) in der nochmals angezeigten Befehlszeile ein Fragezeichen "?" gesetzt. In diesem Fall kann auf die Frage nach Hilfe das HELP-Bild des entsprechenden Befehls angezeigt werden.

Sehr hilfreich ist bei der Fehlerkorrektur der HISTORY-Modus. REDABAS-4 speichert die zuletzt eingegebenen Befehle (Standard=20 oder eine andere, bei SET HISTORY TO <ausdrN> spezifizierte Anzahl). Jeder im HISTORY-Modus gespeicherte Befehl kann zurueckgeholt, korrigiert und wieder mit <ET> zur Ausfuehrung aktiviert werden.
Voraussetzung fuer diese Moeglichkeit ist SET HISTORY ON (Standard). Mit DISPLAY/LIST HISTORY kann der gesamte Inhalt des HISTORY-Speichers angezeigt werden, mit der Pfeiltaste oben (bzw. unten) koennen einzelne Befehle zurueckgeholt werden.

Bei unvollstaendiger Befehlszeile, z.B. fehlenden Dateinamen, kann diese Information nach einer Aufforderung nachgereicht werden.
Weitere Fehler, die bei der Befehlsauswertung zutage treten, zeigt REDABAS-4 mit spezifischen Fehlermeldungen an. Diese

Fehlermeldungen sind selbsterklaerend und beduerfen keiner weiteren Erlaeuterung.

Eine "rollende" Bildschirmausgabe, die eine Seite uebersteigt, kann mit <CTRL-S> angehalten und durch eine beliebige Taste fortgesetzt werden. Die <Esc>-Taste bricht eine Bildschirmausgabe ab.

### 8.4.1.2. Programmodus

REDABAS-4 bietet fuer die Fehlersuche in Programmdateien eine Reihe von Hilfsmoeglichkeiten.
Erkennt REDABAS-4 bei der Abarbeitung einer Programmdatei einen Fehler, erscheint nach der Fehlernachricht

Aufgerufen von - ... (vollstaendiger Name der Programmdatei)

Abbruch, Ignorieren, Unterbrechen ? (A,I,U)

- U=Unterbrechen ermoeglicht die Reaktion auf Fehler, indem temporaer in den interaktiven Modus "umgeschaltet" wird. Die Eingabe von U hat die gleiche Wirkung wie der in einem Programm gegebene Befehle SUSPEND (vgl. anschliessende Beschreibung der Fehlerkorrektur mit SUSPEND).

- Wird A=Abbruch gewaehlt, entspricht das dem Betaetigen der <Esc>-Taste (Voraussetzung ist SET ESCAPE ON!) bzw. dem Befehl CANCEL im Programm.
  Es ist zu beachten, dass ein Abbruch von Programmdateien im allgemeinen Veraenderungen (Arbeitsbereiche, geoeffnete Dateien, SET-Parameter) hinterlaesst, die jegliche weitere Arbeit waehrend dieser REDABAS-4-Sitzung beeinflussen koennen.

- Bei I=Ignorieren wird das Programm weiterhin durchlaufen, im guenstigen Fall bis zum Ende oder aber bis zum naechsten auftretenden Fehler.

Die Moeglichkeit, Fehler im Dialog zu suchen, kann in Programmen mit dem eigens fuer die Fehlersuche konzipierten Befehle SUSPEND genutzt werden. Wird SUSPEND an den Anfang einer fehlerverdaechtigen Programmpassage gesetzt, wird temporaer der interaktive Modus von REDABAS-4 aufgerufen. Mit DISPLAY LIST HISTORY kann der bzw. die den Fehler ausloesenden Befehle ueberprueft werden.
Voraussetzung fuer die Nutzung von DISPLAY/LIST HISTORY in diesem Fall ist, dass SET DOHISTORY auf ON gesetzt wurde (abweichend vom Standard, der OFF ist). Ausserdem kann mit den Befehlen DISPLAY/LIST MEMORY und DISPLAY/LIST STATUS der aktuelle Status abgefragt werden.

Besteht Klarheit ueber verwendete Speichervariablen, aktuellen Arbeitsbereich usw. zum gegenwaertigen Stand, kann mit SET STEP ON (Standard ist OFF!) eine schrittweise Programmabarbeitung erzwungen werden und jeweils ein Befehl ueberprueft und gegebenenfalls korrigiert werden. Es koennen aber auch Programmschalter und andere Bedingungen veraendert werden, um beliebige Programmzweige probeweise abzuarbeiten.

Mittels RESUME kann die Programmausfuehrung wieder dort gestartet werden, wo sie unterbrochen wurde. Wurde der Fehler erkannt und der fehlerhafte Befehl berichtigt, wird SET STEP wieder OFF gesetzt und danach RESUME eingegeben.
Zu beachten ist, dass auf diese Weise am Programm vorgenommene Aenderungen nur temporaer sind. Sollen die Aenderungen in die Programmdatei aufgenommen werden, muss diese mit MODIFY COMMAND oder einem Textprozessor nachtraeglich modifiziert werden.

Hilfestellung bei der Testung eines Programms geben neben SET STEP ON die Befehle SET DEBUG ON, SET ECHO ON, SET TALK ON (vgl. Abschn. 9.3.3.).

REDABAS-4 bietet darueber hinaus Moeglichkeiten fuer das "Abfangen" von Fehlern und Tastatureingaben mit den Befehlen ON ERROR, ON ESCAPE, ON KEY. Der normale Programmablauf wird beim Auftreten eines Fehlers oder bei einem bestimmten Tastendruck unterbrochen und eine bestimmte Reaktion veranlasst. (Eine detaillierte Beschreibung dieser Befehle mit Anwendungsbeispielen ist im Abschn. 9.3.3. zu finden.)

Der gebraeuchlichste Fall fuer das Reagieren auf Fehler wird der Aufruf einer vom Nutzer programmierten Fehlerroutine nach einem ON ERROR-Befehl sein.
Die Fehlernummer kann mit der ERROR()-Funktion erfragt werden, die Fehlernachricht mit MESSAGE(), INKEY() sagt aus, welche Taste zuletzt gedrueckt wurde.

Nach den seitenorientierten Befehlen kann mit READKEY() ueberprueft werden, ob eine Modifizierung des zu editierenden Datensatzes erfolgte, und es kann eine entsprechende Programmreaktion ausgeloest werden.

Anhang A dieser Programmtechnischen Beschreibung bietet eine Auflistung der gebraeuchlichen REDABAS-4-Fehlernachrichten mit der zugehoerigen Nummer.

Hinweise:
---

- Ausserdem gibt es weitere Fehlerzustaende und zugehoerige Nachrichten, die nicht mittels ON ERROR abfangbar sind, sich also einer programmierten Behandlung entziehen.

- Bei der Arbeit mit REDABAS-4 kann es vorkommen, dass ein interner Fehler (z.B. unerkannter Maschinencode) angezeigt wird, wobei ein unmittelbarer Zusammenhang mit dem ausgefuehrten REDABAS-4-Befehl bzw. den behandelten Daten nicht ersichtlich ist. Moegliche Ursachen koennten sein:

  - Ueberschreibungen im Arbeitsspeicher, z.B. durch Ausfuehren unzulaessiger Moduln mit CALL, Rufen fehlerhafter Programme mit RUN oder Verwenden stoerender residenter Programme;

  - Zerstoeren der Programme oder der Steuerdaten von Dateien (z.B. Datenbank- oder Indexdateien) auf externen Datentraegern;

  - Versagen der Hardware.

Entsprechend empfehlen sich u.a. einige der folgenden Massnahmen:

- Beenden und Neustarten von REDABAS-4, falls noch moeglich;
- Warmstart des Betriebssystems, falls moeglich, sonst Kaltstart;
- Erzeugen einer neuen REDABAS-4-Arbeitskopie von der Auslieferungsdiskette;
- Wiederherstellen der Dateien von einer Sicherungskopie;
- Kontrolle der verwendeten externen Programme, des Betriebssystems bzw. der Hardware.

### 8.4.2. Seitenmodus und Cursorsteuerung

Die einfache Bildschirmausgabe im interaktiven Modus erfolgt so, dass der Bildschirm von unten beginnend zeilenweise beschrieben wird. Der Cursor zeigt dabei die naechste freie Bildschirmposition an. Ist eine Bildschirmzeile beschrieben, so rueckt gewoehnlich das ganze Bild eine Zeile nach oben. Ist der Bildschirm gefuellt, so verschwindet die oberste Zeile, und die unterste Bildschirmzeile wird fuer die naechste Anzeige frei. Man bezeichnet dies als "vertikales Rollen des Bildes".

Moderne Bildschirme besitzen die Moeglichkeiten der zeichenweisen Adressierung aller Bildschirmpositionen. Damit koennen die Rechnerprogramme ihre Ein- und Ausgaben auf dem Bildschirm gezielt plazieren. Auch REDABAS-4 nutzt diese Moeglichkeiten. Man nennt diese Betriebsart, waehrend der der gesamte Bildschirm im Zugriff ist, "Seitenmodus" (oder bildschirmorientierte Eingabe). Abweichend von der im interaktiven Modus beschriebenen Bildsschirmausgabe, wird im Seitenmodus der Bildschirm geloescht und zur Anzeige von Menues, Hilfen oder Meldungen benutzt.
Befehle, die diesen Seitenmodus nutzen, sogenannte seitenorientierte Befehle, sind APPEND, ASSIST, BROWSE, CHANGE, CREATE, CREATE/MODIFY LABEL, CREATE/MODIFY QUERY, CREATE/MODIFY REPORT, CREATE/MODIFY SCREEN, CREATE/MODIFY VIEW, EDIT, HELP, INSERT, MODIFY COMMAND, MODIFY STRUCTURE, READ (in Verbindung mit vorangegangenen @...SAY...GET - oder @...GET...-Befehlen und unter der Voraussetzung der Standardeinstellung SET DEVICE TO SCREEEN) und SET.

Im erweiterten Sinne gehoert auch der @-Befehl zu den seitenorientierten Befehlen bezueglich des unmittelbaren Zugriffs auf Bildschirmpositionen. Die Cursortasten der seitenorientierten Befehle sind jedoch fuer @...GET...-Befehle erst nach einem READ wirksam.

Bei den Befehlen APPEND, BROWSE, CHANGE, CREATE, EDIT, INSERT, MODIFY STRUCTURE und READ werden dem Nutzer sogenannte Masken fuer die Eingabe bzw. Aenderung von Daten angeboten. In besonders gekennzeichnete Bereiche (hell unterlegt bzw. invers dargestellt) werden die Daten ueber die Tastatur eingegeben. Mit den Pfeiltasten (bzw. Cursortasten) wird der Cursor in die

gewuenschte Richtung bewegt. Das gilt auch fuer die Befehle, in denen Daten zur Systemsteuerung einzugeben sind, wie z.B. CREATE/MODIFY REPORT.

In der folgenden Uebersicht sind die Tasten bzw. Tastenkombinationen angefuehrt, die im Seitenmodus benutzt werden koennen.

Cursortasten bzw. Tastenkombination im Seitenmodus

| Steuertaste | Alternative | Funktion |
|---|---|---|
| Oben-Pfeiltaste | <CTRL-E> | Bewegen des Cursors um eine Zeile oder ein Feld nach oben bzw. in Menues Markierungsbalken eine Option nach oben |
| Unten-Pfeiltaste | <CTRL-X> | Bewegen des Cursors um eine Zeile oder ein Feld nach unten bzw. in Menues Markierungsbalken eine Option nach unten. Fuer jede Befehlseingabe im interaktiven Modus gilt zusaetzlich: Bewegen des Cursors zum Anfang der Befehlszeile. |
| Links-Pfeiltaste (< <- >) | <CTRL-S> | Bewegen des Cursors um ein Zeichen nach links bzw. in Menues Markierungsbalken eine Option nach links. Zusaetzlich gilt fuer <CTRL-S>: Anhalten und Fortsetzen des vertikalen Rollens des Bildes. |
| Rechts-Pfeiltaste (< -> >) | <CTRL-D> | Bewegen des Cursors um ein Zeichen nach rechts bzw. in Menues Markierungsbalken eine Option nach rechts |
| <CTRL- -> > | <CTRL-B> | Bei CREATE bzw. MODIFY STRUCTURE: Ein Feld nach rechts, wenn Cursor in linker Spalte steht<br>Bei BROWSE: Ein Feld nach rechts (horizontales Blaettern), wenn ein Datensatz die Bildschirmbreite uebersteigt<br>Bei Befehlen zur Behandlung von Merkfeldern und MODIFY COMMAND: Setzen des Cursors an das Ende der Zeile |
| <CTRL- <- > | <CTRL-Z> | Bei CREATE bzw. MODIFY STRUCTURE: Ein Feld nach links, wenn Cursor in rechter Spalte steht<br>Bei BROWSE: Ein Feld nach links (horizontales Blaettern), wenn ein Datensatz die Bildschirmbreite uebersteigt<br>Bei Befehlen zur Behandlung von Merkfeldern und MODIFY COMMAND: Setzen des Cursors an den Anfang der Zeile |

| | | |
|---|---|---|
| <-- | | Loeschen des Zeichens links vom Cursor und Ruecksetzen des Cursors |
| <Del> | <CTRL-G> | Loeschen des Zeichens, auf dem der Cursor steht und Aufruecken von rechts |
| <Home> | <CTRL-A> | Bewegen des Cursors zum Wortanfang bzw. um ein Wort nach links |
| <End> | <CTRL-F> | Bewegen des Cursors zum Wortende bzw. um ein Wort nach rechts |
| <CTRL-End> | <CTRL-W> | Bei APPEND, BROWSE, CHANGE, CREATE, EDIT, INSERT, MODIFY COMMAND, MODIFY STRUCTURE und READ: Beenden des Seitenmodus und Abspeichern der Aenderungen<br>Bei CREATE/MODIFY REPORT: Zoom aus |
| <Esc> | <CTRL-Q> | Beenden des Seitenmodus ohne Abspeichern der Aenderungen und Rueckkehr zum Bereitschaftszeichen<br>Ausnahme: Bei APPEND und BROWSE werden nur die Aenderungen am aktuellen Satz nicht gespeichert<br>Bei ASSIST: Beenden des Seitenmodus aus Pull-down-Menues<br>Bei Untermenues: Ruecksprung ins Hauptmenue<br>Bei HELP und SET: Beenden des Seitenmodus |
| <CTRL-Home> | <F10> | Bei CREATE, BROWSE und MODIFY STRUCTURE:<br>An- und Ausschalten der Menuezeile<br>Bei CREATE/MODIFY LABEL, CREATE/MODIFY QUERY, CREATE/MODIFY REPORT, CREATE/MODIFY SCREEN und CREATE/MODIFY VIEW: An- und Ausschalten von Zusatzmenues<br>Bei CREATE/MODIFY SCREEN: Umschalten zwischen Zeichenbrett und Pull-down-Menues<br>Bei HELP: Ruecksprung zum vorhergehenden Menue<br>Zusaetzlich nur <CTRL-Home> bei APPEND, CHANGE, CREATE, EDIT und INSERT: Eingabe und Editieren von Merkfeldern |
| <F1> | | An- und Ausschalten des Hilfsmenues zur Cursorbewegung<br>Bei ASSIST: An- und Ausschalten des Hilfe-Textes<br>Bei CREATE/MODIFY REPORT: Umschalten zwischen Hilfsmenue und Report-Format |

| | | |
|---|---|---|
| <Ins> | <CTRL-V> | Ein- und Ausschalten des Einfuege-Modus. Im Einfuegemodus wird das einzufuegende Zeichen links von der Cursorstellung eingefuegt |
| <CTRL-K und B> | | Bei Befehlen zur Behandlung von Merkfeldern und MODIFY COMMAND: Reformatieren eines Absatzes |
| <CTRL-K und F> | | Bei Befehlen zur Behandlung von Merkfeldern und MODIFY COMMAND: Erstes Vorfinden der angegebenen Zeichenfolge |
| <CTRL-K und L> | | Bei Befehlen zur Behandlung von Merkfeldern und MODIFY COMMAND: Naechstes Vorfinden der angegebenen Zeichenfolge |
| <CTRL-K und R> | | Bei Befehlen zur Behandlung von Merkfeldern und MODIFY COMMAND: Einlesen einer anderen Datei in die aktuelle Datei |
| <CTRL-K und W> | | Bei Befehlen zur Behandlung von Merkfeldern und MODIFY COMMAND: Schreiben der aktuellen Datei in eine andere Datei |
| <CTRL-N> | | Einfuegen einer neuen Zeile (MODIFY COMMAND) oder Feld- bzw. Spaltendefinition (MODIFY STRUCTURE bzw. CREATE/MODIFY REPORT) |
| <PgUp> | <CTRL-R> | Zurueckgehen auf den vorhergehenden Satz, auf die voherige Bildschirmseite, bei BROWSE zum vorhergehenden "Fenster" oder bei CREATE/MODIFY REPORT zur vorhergehenden Spaltendefinition |
| <CTRL-PgUp> | <CTRL-End> | Sichern und Rueckkehren aus dem Eingeben bzw. Editieren von Merkfeldern. Bei CREATE/MODIFY REPORT: Zoom aus |
| <PgDn> | <CTRL-C> | Weitergehen zum naechsten Satz, zur naechsten Bildschirmseite, bei BROWSE zum naechsten "Fenster" oder bei CREATE/MODIFY REPORT zur naechsten Spaltendefinition |
| <CTRL-PgDn> | <CTRL-Home> | Bei Befehlen zur Behandlung von Merkfeldern:<br>Einsprung in das Eingeben bzw. Editieren von Merkfeldern<br>Nur <CTRL-PgDn> bei CREATE/MODIFY REPORT: Zoom ein |

| | |
|---|---|
| <CTRL-T> | Loeschen des Wortes rechts vom Cursor |
| <CTRL-U> | Bei APPEND, BROWSE, CHANGE, EDIT und INSERT: Setzen der Loeschmarkierung<br>Bei MODIFY REPORT bzw. MODIFY STRUCTURE: Loeschen einer Spalten- bzw. Felddefinition |
| <CTRL-Y> | Loeschen bis Feldende<br>Bei Befehlen zur Behandlung von Merkfeldern und MODIFY COMMAND: Loeschen der aktuellen Zeile<br>Fuer jede Befehlseingabe im interaktiven Modus gilt: Loeschen der Befehlszeile |
| <ET> | Bewegen des Cursors zum naechsten Feld oder zur naechsten Zeile<br>Bei APPEND: Verlassen des Seitenmodus mit Abspeichern der Aenderungen, wenn die erste Position des naechsten leeren Satzes erreicht ist<br>Bei EDIT: Verlassen des Seitenmodus mit Abspeichern der Aenderungen, wenn man sich am letzten Feld des letzten Satzes befindet<br>Bei Befehlen zur Behandlung von Merkfeldern und MODIFY COMMAND: Einfuegen einer Zeile, wenn "Einfuegen" eingeschaltet ist<br>In Menues wird mit <ET> die voreingestellte Option ausgewaehlt oder ein fester Wert eingestellt. |

Anmerkung: Zur Ausgabe von Ergebnissen ueber Drucker kann der Drucker durch <CTRL-P> vor der Befehlseingabe eingeschaltet und nach der Befehlsausfuehrung ausgeschaltet werden.

## 8.5. Zugriffspfade

Zu den wesentlichen Leistungen des Betriebssystems DCP gehoert das Verwalten einer Verzeichnishierarchie auf jedem Datentraeger (Festplatte oder Diskette), das Einstellen und Registrieren eines aktuellen Verzeichnisses je Laufwerk und das Akzeptieren von Zugriffspfaden beim Ansprechen von Dateien. Diese Moeglichkeiten, die besonders bei Datentraegern hoher Kapazitaet grosse Vorzuege bieten, koennen Anfaengern einige Schwierigkeiten bereiten. Das Pfadkonzept des DCP wirkt sich einerseits auf die Anwendungsbedingungen von Softwareprodukten fuer DCP aus, wird andererseits von solchen Produkten haeufig fuer eigenstaendige Verwaltungsleistungen genutzt. Bei REDABAS-4 hat das Pfadkonzept sowohl fuer das Anlegen und Ansprechen aller verwalteten Dateien als auch fuer die Gewaehrleistung des Zugriffs auf die REDABAS-4-Systemkomponenten selbst Bedeutung.

### 8.5.1. Aufruf der REDABAS-4-Systemkomponenten

Wenn REDABAS-4 nur unter dem Programmnamen seiner Startkomponente (REDABAS), also ohne Laufwerks- oder Pfadangabe, aufgerufen wird, dann gilt die im DCP gewaehlte Einstellung: Zunaechst wird im aktuellen Verzeichnis des aktuellen Laufwerks nach den Programmkomponenten gesucht. Bleibt diese primaere Suche ohne Erfolg, werden noch alle die Verzeichnisse durchgegangen, die mit den durch das PATH-Kommando des DCP eingefuehrten zusaetzlichen Suchpfaden ausgewaehlt sind.

Nach dem geschilderten Prinzip erfolgt auch die Suche nach der Konfigurierungsdatei REDAB.CF von REDABAS-4 (siehe Abschnitt 10.1), und nach der Umkodierungstabelle REDABP.TB fuer die Druckausgabe (siehe Abschnitt 10.2.). Beide Dateien sind keine notwendigen Systembestandteile, sondern koennen wahlweise bereitgestellt werden. Wird REDAB.CF nicht gefunden, arbeitet REDABAS-4 mit seinen Standardeinstellungen, wird REDABP.TB nicht gefunden, findet keine Umkodierung der Druckausgabe statt.

Auch beim Aufruf eines externen DCP-Kommandos oder Anwendungsprogramms aus REDABAS-4 heraus mit

```
RUN <kommando> ...
```

laeuft der beschriebene Suchvorgang ab. Allgemein ist aber die Angabe von Laufwerk und/oder Pfad moeglich:

```
RUN [<laufwerk>][<pfad>\]<kommando> ...
```

Diese zusaetzlichen Angaben werden nicht durch REDABAS-4 verarbeitet, sondern durch das Betriebssystem DCP. Die Suche nach der das <kommando> realisierenden Programmkomponente laeuft ab, als waere ausserhalb REDABAS-4 der Aufruf

```
[<laufwerk>][<pfad>\]<kommando>
```

durch das Betriebssystem zu verarbeiten:

Ist <laufwerk> angegeben, gilt dieses Laufwerk statt des (im DCP eingestellten) aktuellen Laufwerks. Ohne Pfadangabe <pfad>\ ist das aktuelle Verzeichnis dieses Laufwerks fuer die <kommando>-Suche massgebend. Ein spezifizierter Pfad kann vollstaendig formuliert sein (beginnend mit \, also ab dem Stammverzeichnis des Datentraegers) oder relativ auf das eingestellte aktuelle Verzeichnis des geltenden Laufwerks bezogen sein.

Eine explizite Pfadangabe verhindert ausserdem, dass im Falle erfolgloser Suche die ueber PATH dem DCP bekanntgegebenen zusaetzlichen Suchpfade mit herangezogen werden. Weitere Einzelheiten sind der Dokumentation zum DCP zu entnehmen.

Analog ist auch REDABAS-4 selbst ueber Laufwerks- und/oder Pfadangabe aufrufbar:

[<laufwerk>][<pfad>\]REDABAS

Erfolgreich ist ein solcher Aufruf nur nach besonderen Vorkehrungen, da REDABAS-4 aus mehreren Programmkomponenten besteht. Waehrend der Arbeit laedt es Teile von Programmkomponenten nach und fuehrt Ueberlagerungen aus, um den Arbeitsspeicher rationell zu nutzen. Angaben zu Laufwerk und/oder Pfad beim REDABAS-4-Aufruf wuerden nur die Suche der Startkomponente beeinflussen. Fuer alle anderen Komponenten gilt aber weiter das Suchprinzip, das eingangs dieses Abschnitts beschrieben wurde, also erst aktuelles Verzeichnis des aktuellen Laufwerks, dann PATH-Angaben.

Im Abschnitt 8.5.3. werden einige sinnvolle Arbeitsweisen als Beispiel erlaeutert.

### 8.5.2. Dateizugriff in REDABAS-4

Aehnliche Moeglichkeiten, wie das Betriebssystem DCP fuer den Aufruf seiner Kommandos bietet, realisiert REDABAS-4 fuer den Zugriff auf die Dateien, die es verwaltet: Sowohl ein aktuelles Laufwerk (Standardlaufwerk) als auch eine Folge von Suchpfaden ist angebbar.

Das aktuelle Laufwerk fuer Dateien (z.B. Datenbankdateien, Indexdateien) wird, wie bereits im Abschnitt 2.3.2. erwaehnt, mittels

SET DEFAULT TO <laufwerk>

eingestellt. Zusaetzliche Suchpfade fuer existierende Dateien sind mit dem REDABAS-4-Befehl

SET PATH TO <pfadfolge>

definierbar. Eine Moeglichkeit zur Voreinstellung dieser beiden Parameter bietet auch die Konfigurierungsdatei REDAB.CF (siehe Abschn.10.1.).

Ausserdem kann an den Stellen in der REDABAS-4-Befehlssprache, an denen eine Datei anzugeben ist (<db-datei> in USE, <indexdatei> in INDEX usw., hier einfach mit <datei> bezeichnet,) gewoehnlich auch ein Laufwerk und/oder ein Pfad explizit angegeben werden, also

[<laufwerk>][<pfad>\]<datei>

anstelle von <datei>. Damit ergibt sich eine grosse Zugriffsvielfalt, die im folgenden systematisch erklaert werden soll.

Laufwerksauswahl:

- Ein bei der Dateiangabe explizit aufgefuehrtes Laufwerk (kuenftig als ld abgekuerzt) hat Prioritaet.
- Das im DCP eingestellte aktuelle Laufwerk (kuenftig als la abgekuerzt) ist zustaendig, wenn weder SET DEFAULT gegeben wurde noch bei der Dateiangabe explizit ein Laufwerk aufge-

fuehrt wird.

- Das beim letzten SET DEFAULT-Befehl benannte Laufwerk (kuenftig als ls abgekuerzt) ist zustaendig, wenn bei der Dateiangabe kein Laufwerk explizit angegeben wurde.

Verzeichnisauswahl:

- Wenn bei der Dateiangabe explizit ein Pfad mitgeteilt wird, bestimmt dieser Pfad das Verzeichnis im ausgewaehlten Laufwerk ld oder ls oder la (siehe oben). Der Zugriffspfad kann dabei vollstaendig, d.h. ab Stammverzeichnis des Datentraegers, oder relativ zum aktuellen Verzeichnis dieses Laufwerks angegeben werden (siehe DCP-Dokumentation).
  Falls der REDABAS-4-Befehl an dieser Stelle eine bereits existierende Datei verlangt, die Suche im ausgewaehlten Verzeichnis aber erfolglos blieb, wird ein Fehler gemeldet. Durch SET PATH eingefuehrte zusaetzliche Suchpfade werden also ignoriert, wenn bei der Dateiangabe explizit ein Pfad aufgefuehrt ist.

- Wird bei der Dateiangabe kein Pfad explizit aufgefuehrt, gilt primaer das aktuelle Verzeichnis des ausgewaehlten Laufwerks ld oder ls oder la (in Analogie zum DCP).
  Falls der REDABAS-4-Befehl an dieser Stelle eine bereits existierende Datei verlangt, wird die Suche in den Verzeichnissen fortgesetzt, die durch den letzten REDABAS-4-Befehl SET PATH ueber ihre Zugriffspfade spezifiziert wurden.

Hinweise:

- Ein Zugriffspfad legt (in Analogie zum DCP) immer nur ein einzelnes Verzeichnis fest, d.h. bei mehrstufigen Pfadangaben werden nicht etwa die Verzeichnisse aller uebergeordneten Stufen mit durchsucht.

- Die Bezugnahme auf eine dem aktuellen Verzeichnis uebergeordnete Stufe mittels .. ist in REDABAS-4 aus syntaktischen Gruenden nicht in jedem Falle moeglich.

- Zusaetzliche Suchpfade, die durch das DCP-Kommando PATH eingefuehrt wurden, werden bei der Suche nach Dateien in REDABAS-4 grundsaetzlich ignoriert (ihre Bedeutung fuer REDABAS-4 beschreibt Abschn. 8.5.1).

- Die REDABAS-4-Befehle SET DEFAULT und SET PATH haben keinen Einfluss auf die entsprechenden Einstellungen (aktuelles Laufwerk und zusaetzliche Suchpfade) des DCP. Sie gelten nur REDABAS-4-intern fuer die durch REDABAS-4 verwalteten Dateien.

Ergaenzungen zu SET PATH:

- Diese Pfade koennen nur bei der Suche nach vorhandenen Dateien zum Zwecke des Lesens oder Aenderns Bedeutung haben (z.B. Befehl TYPE), nie aber beim Anlegen einer neuen Datei. Z.B. sucht MODIFY COMMAND prog nach erfolgloser primaerer Suche alle durch zusaetzliche Zugriffspfade definierten Verzeichnisse nach prog.PRG ab, solange diese Datei nicht gefunden ist. Wird die Datei aufgefunden, wird auch ihr neuer

Stand nach der Aenderung ins Ursprungsverzeichnis zurueckgebracht. Andernfalls wird prog.PRG im aktuellen Verzeichnis des ausgewaehlten Laufwerks ld oder ls oder la (siehe oben!) neu angelegt.

- Den aufgefuehrten Pfaden sollte moeglichst das Laufwerk vorangestellt werden, ausserdem ist eine vom Stammverzeichnis ausgehende vollstaendige Pfadangabe angeraten. Ohne Laufwerksangabe ist das im DCP aktuelle Laufwerk der Bezug, nicht etwa die Laufwerksangabe bei SET DEFAULT. Unvollstaendige Pfadangaben wirken relativ zum aktuellen Verzeichnis des entsprechenden Laufwerks.

Zusammenfassung zur Auswahl von Laufwerk und Verzeichnis:

| Dateiangabe | ohne SET DEFAULT | mit SET DEFAULT TO ls |
|---|---|---|
| <datei> | aktuell in la [+...] | aktuell in ls [+...] |
| ld:<datei> | aktuell in ld [+...] | aktuell in ld [+...] |
| pfad\<datei> | Verzeichnis in la | Verzeichnis in ls |
| ld:pfad\<datei> | Verzeichnis in ld | Verzeichnis in ld |

[+...] soll die bei Bedarf stattfindende Bezugnahme auf die mit SET PATH eingefuehrten Verzeichnisse ausdruecken

Besonderheit beim Befehl DIR:
Dieser Befehl erfuellt in REDABAS-4 neben dem allgemeinen Anzeigen des Verzeichnisinhalts eine datenbankspezifische Sonderfunktion: Das Anzeigen der Datenbankdateien (Typ.DBD) mit ergaenzenden Informationen. Das gilt, falls entweder kein Dateinamen-Muster oder keine Erweiterungsangabe eingegeben wurde. Aus diesem Grunde ist DIR [<laufwerk>]<pfad> im Unterschied zum entsprechenden DCP-Kommando in REDABAS-4 nicht gueltig, dagegen liefert dieser Befehl in der folgenden Formulierung

DIR [<laufwerk>]<pfad>\

Anzeigen zu allen DBD-Dateien aus dem angesteuerten Verzeichnis (d.h. der Pfad muss auch mit \ abgeschlossen werden, wenn keine weitere Angabe folgt).

### 8.5.3. Beispiele zu den Arbeitsmoeglichkeiten

Fuer alle Beispiele wird die folgende Umgebung vorausgesetzt:
Ein REDABAS-4-Anwendungskomplex besteht aus Anwendungsfaellen "ANW1" bis "ANW4" und allgemeingueltigen Dateien (z.B. im gesamten Komplex haeufig benutzte Datenbankdateien, Programmdateien usw.). Die REDABAS-4-Systemkomponenten befinden sich auf einer Festplatte, Laufwerk C.

Festplatte in C:
\REDABAS : Systemkomponenten von REDABAS-4
\REDABAS\ALLG : Allgemeingueltiges zum Anwendungskomplex (hier sollen keine neuen Dateien angelegt werden)
\REDABAS\ANW1 : Anwendungsfall 1 (haeufigster Anwendungsfall)

```
Disketten:
#1 \(Stammverzeichnis): Anwendungsfall 2
#2 \ANW3               : Anwendungsfall 3
   \ANW4               : Anwendungsfall 4

Beispiel 1:
-----------
```

Anwendungsfall 2 ist zu bearbeiten, d.h. die Diskette #1 ist in Laufwerk A einzulegen, und alle neu entstehenden Dateien sind auf dieser Diskette anzulegen.

- Im DCP wird C als aktuelles Laufwerk eingestellt und \REDABAS zum aktuellen Verzeichnis erklaert.

- Aufrufen von REDABAS-4 (C:\REDABAS>REDABAS) und Erklaeren von A zum aktuellen REDABAS-4-Laufwerk mittels

```
SET DEFAULT TO A
```

  Damit wird die Diskette in A sowohl zum Suchen nach vorhandenen Dateien als auch zum Anlegen neuer Dateien aller Typen benutzt, ohne dass vor den Dateinamen die explizite Laufwerksangaben A noetig waere.

```
SET PATH TO C:\REDABAS\ALLG
```

  ermoeglicht zusaetzlich, auch alle im Unterverzeichnis ALLG befindlichen Bestandteile des Komplexes ueber ihren blossen Dateinamen anzusprechen. (Bei Namensgleichheit eines Bestandteils in A und in C:\REDABAS\ALLG hat A Vorrang, so dass gegebenenfalls fuer diese Ausnahmesituation eine genaue Pfadspezifikation noetig ist.)

```
Beispiel 2:
-----------
```

Anwendungsfall 4 ist zu bearbeiten, d.h. Diskette #2 ist in Laufwerk A einzulegen (Im praktischen Betrieb duerfte es aus Kapazitaetsgruenden kaum moeglich sein, eine Diskette aufzuteilen, aber in der Schulungs- und Einarbeitungsphase wird es so ueberhaupt moeglich, viele kleine Dateien oekonomisch unterzubringen.).

- Im DCP wird am einfachsten fuer das (nicht-aktuelle!) Laufwerk A das gewuenschte Unterverzeichnis \ANW4 zum aktuellen erklaert, also

```
C:\REDABAS>CHDIR  A:\ANW4
```

- Weiter wie bei Beispiel 1, d.h. nach SET DEFAULT TO A wird automatisch nicht das Stammverzeichnis der Diskette, sondern das vorgewaehlte aktuelle Verzeichnis angesprochen (voellig analog zu den Standards des DCP).

```
Beispiel 3:
-----------
```

Anwendungsfall 1 ist zu bearbeiten

- Im DCP wird C als aktuelles Laufwerk eingestellt und \REDABAS zum aktuellen Verzeichnis erklaert.

- Aufrufen von REDABAS-4 (C:\REDABAS>REDABAS) und Definieren zusaetzlicher Datei-Suchpfade

SET PATH TO C:\REDABAS\ANW1;C:\REDABAS\ALLG

Dafuer eruebrigt sich SET DEFAULT TO C, weil C bereits vom DCP her aktuell ist.
Alle bestehenden Dateien in ANW1 und ALLG sind im Zugriff fuer Lese- und Aenderungsoperationen. Neue Dateien entstehen allerdings, falls (wie z.B. bei INDEX TO inl...) keine explizite Pfadangabe eingegeben wird, im aktuellen Verzeichnis \REDABAS, nicht aber in \REDABAS\ANW1. Das Anlegen neuer Dateien muesste immer mit Sorgfalt geschehen,

z.B. INDEX TO ANW1\inl...
oder INDEX TO \REDABAS\ANW1\inl...

was erfahrungsgemaess zu oft versaeumt wird.

Beispiel 4:
---

Anwendungsfall 1 ist zu bearbeiten, ohne die im Beispiel 3 auftretenden Nachteile beachten zu muessen.

- Im DCP wird C als aktuelles Laufwerk eingestellt und \REDABAS\ANW1 zum aktuellen Verzeichnis erklaert.
  Zusaetzlich wird durch

  PATH ...;C:\REDABAS

  einer der DCP-Suchpfade fuer Programme auf das Verzeichnis C:\REDABAS gerichtet, das die REDABAS-4-Systemkomponenten enthaelt.

- Aufrufen von REDABAS-4 (C:\REDABAS\ANW1>REDABAS) und Definition eines zusaetzlichen Datei-Suchpfads auf die allgemeinen Bestandteile des Anwendungskomplexes durch

  SET PATH TO C:\REDABAS\ALLG

Dann ist weder SET DEFAULT erforderlich, noch eine explizite Pfadangabe bei Dateireferenzen. Alle neu entstehenden Dateien werden automatisch in \REDABAS\ANW1, dem vom DCP her aktuellen Verzeichnis angelegt.

## 8.6. Uebersichten zur Datensatzzeigerverwaltung

### 8.6.1. Bedingungen bei nicht-indizierten Datenbankdateien

Fuer die nachfolgend angefuehrten Befehle bzw. Befehlsfolgen gelten fuer nicht-indizierte Datenbankdateien folgende Zustaende:

| Befehl | RECNO() | EOF() | BOF() | REDABAS-4 Nachricht |
|---|---|---|---|---|
| GO TOP | 1 | .F. | .F. | Nein |
| GO TOP;SKIP -n | 1 | .F. | .T. | Nein |
| GO TOP;SKIP -n;SKIP m | m+1 | .F. | .F. | Nein |
| GO TOP;SKIP -n;SKIP -m | 1 | .F. | .T. | Ja |
| GO BOTTOM | max | .F. | .F. | Nein |
| GO BOTTOM;SKIP | max+1 | .T. | .F. | Nein |
| GO BOTTOM;SKIP n | max+1 | .T. | .F. | Nein |
| GO BOTTOM;SKIP n;SKIP-m | max-m+1 | .F. | .F. | Nein |
| GO BOTTOM;SKIP n;SKIP m | max+1 | .T. | .F. | Ja |
| LIST | max+1 | .T. | .F. | Nein |
| LOCATE (Satz gefunden) | n | .F. | .F. | Nein |
| LOCATE (Satz nicht gefunden) | max+1 | .T. | .F. | Ja |

RECNO() = Satznummer vergl. Abschn. 9.2. RECNO()-Funktion
EOF() = Dateiende vergl. Abschn. 9.2. EOF()-Funktion
BOF() = Dateianfang vergl. Abschn. 9.2. BOF()-Funktion
max = hoechste in der Datenbankdatei vorkommende Datensatznummer, d.h. die Nummer des physisch letzten Datensatzes

## 8.6.2. Bedingungen bei indizierten Datenbankdateien

Fuer die nachfolgend angefuehrten Befehle bzw. Befehlsfolgen gelten fuer indizierte Datenbankdateien folgende Zustaende:

| Befehl | RECNO() | EOF() | BOF() | REDABAS-4 Nachricht |
|---|---|---|---|---|
| GO TOP | erste | .F. | .F. | Nein |
| GO TOP;SKIP -n | erste | .F. | .T. | Nein |
| GO TOP;SKIP -n;SKIP m | erste+m | .F. | .F. | Nein |
| GO TOP;SKIP -n;SKIP -m | erste | .F. | .T. | Ja |
| GO BOTTOM | letzte | .F. | .F. | Nein |
| GO BOTTOM;SKIP | max+1 | .T. | .F. | Nein |
| GO BOTTOM;SKIP n | max+1 | .T. | .F. | Nein |
| GO BOTTOM;SKIP n;SKIP-m | letzte - (m-1) | .F. | .F. | Nein |
| GO BOTTOM;SKIP n;SKIP m | max+1 | .T. | .F. | Ja |
| LIST | max+1 | .T. | .F. | Nein |
| LOCATE (Satz gefunden) | n | .F. | .F. | Nein |
| LOCATE (Satz nicht gefunden) | max+1 | .T. | .F. | Ja |
| FIND (Satz gefunden) | n | .F. | .F. | Nein |
| FIND (Satz nicht gefunden) | max+1 | .T. | .F. | Ja |
| SEEK (Satz gefunden) | n | .F. | .F. | Nein |
| SEEK (Satz nicht gefunden) | max+1 | .T. | .F. | Ja |

RECNO(), EOF(), BOF() vgl. Abschnitt 8.6.1.

erste = nicht die niedrigste in der Datenbankdatei auftretende Satznummer, sondern die Satznummer des laut Indexordnung logisch ersten Satzes

letzte = nicht die hoechste in der Datenbankdatei auftretende Satznummer, sondern die Satznummer des laut Indexordnung logisch letzten Satzes

erste+m = Nummer des Datensatzes, der in der Indexordnung m Saetze nach dem ersten folgt

letzte-m = Nummer des Datensatzes, der in der Indexordnung m Saetze vor dem letzten steht

letzte-(m-1) = Nummer des Datensatzes, der in der Indexordnung (m-1) Saetze vor dem letzten steht

max+1 = Zahl, die um 1 groesser ist als die hoechste in der Datenbankdatei vorkommende Datensatznummer

### 8.6.3. Bedingungen bei SET DELETED OFF/ON

| Befehl | SET DELETED OFF (kein Ausblenden) | SET DELETED ON (mit Ausblenden) |
|---|---|---|
| GO TOP | (logisch) erster Satz | (logisch) erster gueltiger Satz; sind keine gueltigen Saetze vorhanden, wird EOF auf .T. gesetzt |
| GO BOTTOM | (logisch) letzter Satz | (logisch) letzter gueltiger Satz; sind keine gueltigen Saetze vorhanden, werden BOF und EOF auf .T. gesetzt |
| GOTO | angegebener Satz | angegebener Satz |
| SKIP n | ausgehend vom aktuellen Satz n Saetze weiter vor (wenn n positiv) oder zurueck (wenn n negativ) | ausgehend vom aktuellen Satz n gueltige Saetze weiter vor (wenn n positiv) oder zurueck (wenn n negativ) |
| FIND<br>SEEK<br>LOCATE | erster gefundener Satz entsprechend Bedingung | erster gefundener gueltiger Satz entsprechend Bedingung |
| DELETE | Setzen der Loeschmarkierung auf angegebene Saetze, Satzzeiger wird auf letzten geloeschten Satz gestellt | Setzen der Loeschmarkierung auf angegebene Saetze, Satzzeiger wird auf letzten geloeschten Satz gestellt |
| COPY<br>COUNT<br>DISPLAY<br>EDIT<br>RECALL<br>REPLACE<br>SORT<br>SUM | alle Saetze entsprechend <bereich> bzw. bis EOF | alle gueltigen Saetze entsprechend <bereich> bzw. bis EOF |
| INDEX<br>REINDEX | alle Saetze | alle Saetze |